SLOVENSKÁ POĽNOHOSPODÁRSKA UNIVERZITA
V NITRE
Rektor: prof. Dr. Ing. Imrich Okenka, PhD.
MECHANIZAČNÁ FAKULTA
Dekan: prof. Ing. Vladimír Kročko, CSc.

Návrh experimentálneho pracoviska na výučbu mikroprocesorovej techniky

Diplomová práca

Katedra elektrotechniky a automatizácie
Vedúci katedry: Ing. Ondrej Lukáč, PhD.
Vedúci práce: doc. Ing. Dušan Hrubý, PhD.

Marián Plačko

Nitra, 2005

Abstrakt

Diplomová práca sa zaoberá návrhom experimentálneho pracoviska na výučbu mikroprocesorovej techniky. Za cieľ sme si stanovili: jednoduchosť konštrukcie, finančnú nenáročnosť, rýchly vývoj vrátane testovania a využitie na pedagogické účely.

V prvom kroku sme sa zamerali na výber vhodnej architektúry (CISC, RISC), výber mikrokontroléra (ATMEL, MICROCHIP, MOTOROLA a ďalšie) a v neposlednom rade i na výber vhodného programovacieho jazyka (Assembler, C, Visual Basic, Pascal, Java, popr. iné).

V ďalšej časti sme sa zamerali na vlastnú realizáciu experimentálneho pracoviska (hardvéru a softvéru pre mikrokontrolér), pričom sme zohľadnili informácie získané v predchádzajúcej časti. Nadobudnuté informácie sa prejavili vo výbere ako: architektúra − RISC, mikrokontrolér − ATMEL AVR AT90S8535, vyšší programovací jazyk − jazyk C.

Výsledkom nášho zámeru je vývojové pracovisko, ktoré pozostáva z niekoľkých funkčných aplikácií, napr.: maticová klávesnica, LCD, komunikácia s PC prostredníctvom rozhrania RS232, meranie napätia − zbernica MicroWire, snímanie teploty − zbernica 1-Wire a pod.

Kľúčové slová: mikrokontrolér, ATMEL AVR, programovací jazyk C, experimentálne pracovisko, výučba mikroprocesorovej techniky

Abstract

The aim of the graduate thesis is to design a new experimental workstation witch can be used for teaching of microprocessor techniques. Our main objectives are: simplicity of construction, financial appropriateness, quick development, testing and exploitation for pedagogical purposes.

At the beginning we focused on the choice of: suitable architecture (CISC, RISC), microcontroller (ATMEL, MICROCHIP, MOTOROLA, etc.) and appropriate programming language (Assembler, C, Visual Basic, Pascal, Java, etc.).

Then we aimed at the establishment of the experimental workstation (hardware and software for the microcontroller). We took into account the information obtained from the first part of our experiment. We used obtained information to choose the suitable: architecture − RISC, microcontroller − ATMEL AVR AT90S8535 and higher programming language − language C.

Our intention was to establish a workstation witch consists of several functional applications, e.g. a matrix keyboard, LCD, communication with PC via interface RS232, measurement of voltage − bus MicroWire, temperature sensing − bus 1-Wire, etc.

Key words: microcontroller, ATMEL AVR, programming language C, experimental workstation, teaching of microprocessor techniques

Abstrakt

Die Diplomarbeit beschäftigt sich mit dem Vorschlag von experimentell Arbeitsplatz für den Unterricht von Mikroprozessortechnik. Ziel unserer Arbeit ist: die Feststellung der einfachen Konstruktion, der finanziellen Anspruchslosigkeit, der schnellen Entwicklung einschließlich des Testen und Ausnutzung in der pädagogischen Täglichkeit.

Bei dem ersten Schritt orientierten wir uns auf die geeignete Auswahl der Architektur (CISC, RISC), Mikrokontroller (ATMEL, MICROCHIP, MOTOROLA, und so weiter), Programmiersprache (Assembler, C, Visual Basic, Pascal, Java, u.a).

In dem Nächsten Teil orientierten wir uns auf die eigene Durchführung vom experimentellen Arbeitsplatz (Hardware und Software für der Mikrokontroller), dabei berücksichtigten wir die Informationen, die wir dies dem vorigen Teil gewonnen haben. Die gewonnenen Informationen wirkten bei der Auswahl als: Architektur − RISC, Mikrokontroller − ATMEl AVR AT90S8535, höhere Programmiersprache − Sprache C.

Das Ergebnis unserer Untersuchungen ist der Entwicklungsarbeitsplatz, der aus einigen Funktionsapplikationen, z.B.: matrixe Tastatur, LCD, Kommunikation mit PC vermittels Interface RS232, Messung der Spannung − der MicroWire Bus, Messung der Temperatur − der 1-Wire Bus besteht und so weiter.

Die Schlüsselwörter: der Mikrokontroller, ATMEL AVR, die Sprache C, der Experimentale Arbeitsplatz , der Unterricht von der Mikroprozessortechnik

ČESTNÉ VYHLÁSENIE

Podpísaný Marián Plačko vyhlasujem, že som záverečnú prácu na tému "Návrh experimentálneho pracoviska na výučbu mikroprocesorovej techniky" vypracoval samostatne s použitím uvedenej literatúry.

Som si vedomý zákonných dôsledkov v prípade, ak hore uvedené údaje nie sú pravdivé.

V Nitre dňa 30.4.2005

.....

POĎAKOVANIE

Touto cestou si dovoľujem úprimne poďakovať vedúcemu diplomovej práce doc. Ing. Dušanovi Hrubému, PhD. za odborné vedenie, cenné rady, pripomienky, pomoc a ochotu pri vypracovaní mojej diplomovej práce.

Použité označenie

µVision 2 − softvér na programovanie mikrokontrolérov ATMEL

1-Wire − sériová jednovodičová zbernica, Dallas Semiconductor

A/D − (Analog/Digital), analógovo digitálny prevodník

ACSR − register (A/D) mikrokontroléra ATMEL AVR AT90S8535

ALU − (Arithmetic and Logic Unit), aritmetická a logická jednotka

ANSI C − kompilátor jazyka C, štandard

ASM − programovací jazyk

AVR − vývojový rad mikrokontrolérov ATMEL

BOOTLOADER − zavádzač k programovaniu pamäti FLASH a EEPROM

C − vyšší programovací jazyk

CISC − (Complex Instruction Set Computer), architektúra mikrokontrolérov

CLK − (Clock), hodinový signál

CMOS − technológia (so zníženou spotrebou napájania) elektronických súčiastok

CodeVisionAVR C − softvér na programovanie mikrokontrolérov ATMEL AVR v jazyku C

CPU − (Central Processor Unit), centrálna procesorová jednotka

CVS − program na vývoj aplikácii v jazyku C

D/A − (Digital/Analog), digitálne analógový prevodník

DEBUGGER − nástroj na odladťovanie a ladenie programov

DIP − puzdro integrovaného obvodu

EEPROM − elektricky programovateľná pamäť, vymazáva sa UV žiarením

EMBEDDED − vstavané zariadenie

EPROM − elektricky programovateľná a zmazateľná pamäť

ERRDATA − oprava chybných dát v datascheet-och

FIRMWARE − mikroprogramové vybavenie

FLASH − elektricky programovateľná a zmazateľná pamäť

GND − zem napájania

GNU − druh licencie

HS − (High Speed), režim pre kryštálový oscilátor, mikrokontroléry PIC

I/O − (Input/Output), vstup/výstup

I^2C − (Inter-Integrated-Circuit Bus), sériová 2 vodičová zbernica, Philips

IAP − (In-Application Programming), umožňuje aktualizáciu firmvéru bez nutnosti prerušiť prácu zariadenia

IAR − softvér na programovanie mikrokontrolérov ATMEL

IDE − (Integrated Development Environment), integrované vývojové prostredie

IDLE mód − mód mikrokontroléru, v ktorom je zastavená CPU, ale čítače/časovače, Watchdog a prerušovací systém neprestávajú pracovať

IRC − režim oscilátora, mikrokontroléry PIC, kalibrovaný RC oscilátor integrovaný na čipe

ISP − (In-System Programming), metóda, umožňuje programovanie obvodov priamo v zariadení

JAVA − vyšší programovací jazyk

JTAG − (Joint Test Actoin Group), rozhranie pre diagnostiku a testovanie integrovaných obvodov

Keil C51 − softvér na programovanie mikrokontrolérov ATMEL

LCD − (Led Crystal Dispaly), zobrazovacia jednotka

LF − (Low Frequency), režim s nízkym odberom, kryštálový oscilátor, mikrokontroléry PIC

LP − (Low Power), režim s nízkym odberom, kryštálový oscilátor, mikrokontroléry PIC

LPM − (Load Program Memory), čítanie/zápis pamäti pri Bootloader-y

LPT − paralelný port PC

MASTER − nadriadené zariadenie

MCU − (microcontroller), mikrokontrolér

MicroWire − sériová trojvodičová zbernica, National Semiconductor

MIPS − (Million Instructions Per Second), výkon mikrokontroléra

MISO − vodič pre sériový čítanie obsahu interných pamätí FLASH a EEPROM

MLF − puzdro integrovaného obvodu

MOSI − vodič pre sériový zápis obsahu interných pamätí FLASH a EEPROM

NITRON − mikrokontrolér od firmy MOTOROLA

OPCODE − 16-bitová zbernica pre mikrokontroléry ATMEL AVR

OTP − pamäť programovateľná len jedenkrát

OVERDRIVE − druh komunikácie (prevod do rýchla) u zbernice 1-Wire

PASCAL − programovací jazyk

PC − (personal computer), osobný počítač

PDIP − puzdro integrovaného obvodu

PIC − mikrokontroléry firmy MICROCHIP

PIPELINE − jednostupňová zbernica u mikrokontrolérov ATMEL AVR ktorá umožňuje, aby sa počas výkonu inštrukcie následná inštrukcia prenášala z pamäte

PLCC − puzdro integrovaného obvodu

PLL − syntezátor frekvencie

POWER DOWN − režim "spánku" mikrokontroléra, v ktorom je zastavený i externý oscilátor

PQFP − puzdro integrovaného obvodu

PSW − register mikrokontroléra

PWM − impulzne šírková modulácia

RAM − (Read Access Memory) elektronická pamäť, ktorá sa po vypnutí napájania vymaže

RC − režim oscilátora u mikrokontrolérov PIC

RD − riadiaci signál (čítanie) pre správu vonkajšej programovej pamäti u x51

RISC − (Reduced Instruction Set Controllers), architektúra mikrokontroléra

ROM − elektronická pamäť, v ktorej po vypnutí napájania zostávajú informácie zachované

RTC − (Real Timer Clock), hodiny reálneho času

RUN mód − mód, v ktorom všetky časti mikrokontroléra v prevádzke

SC − (Chip Select), výber periférneho obvodu u zbernice MicroWire

SCK − vodič pre synchronizáciu dát

SDA − vodič slúžiaci na synchronizačný prenos dát u zbernice I^2C

SDCC − (Small Device C Compiller), kompilátor s licenciou GNU

SFR − register mikrokontroléra

SLAVE − podriadené zariadenie

SLEEP mód − úsporný režim mikrokontroléra

SO/DI − (Serial Out/Data In), prepojuje výstup mikrokontroléra so vstupmi periférneho zariadenia a naopak

SOIC − puzdro integrovaného obvodu

SPI − (Serial Peripheral Interface) rozhranie pre sériovú synchrónnu komunikáciu

SPM − (Store Program Memory), čítanie/zápis pamäti pri Bootloader-y

SRAM − (Static Access Read Memory)

SSOP − puzdro integrovaného obvodu

STACK − zásobník

START KIT − vývojová doska (hardvér)

SW − (softvér), programové vybavenie

TQFP − puzdro integrovaného obvodu

TTL − číslicová logika

UART − (Universal Asynchronous Receiver Transmitter), rozhranie pre sériovú asynchrónnu komunikáciu

Ucc − napájacie napätie

VDD − napájacie napätie

Visual Basic − vyšší programovací jazyk

WATCHDOG − sledovač činností

WIZARD − SW formulár pre rýchlejšiu a efektívnejšiu prácu

WR − riadiaci signál (zápis) pre správu vonkajšej programovej pamäti u x51

x51 − mikrokontroléry firmy ATMEL, založené na vývojovom rade 8051 firmy INTEL

XRAM − pamäťový model

XT − režim pre kryštálový oscilátor, mikrokontroléry PC

Obsah

ÚVOD

Mikroprocesory spôsobili vo vývoji elektroniky doslova revolúciu rovnako ako v minulosti prvý tranzistor. Vývoj postupoval od najjednoduchších 4-bitových až k dnešným 32-bitovým resp. 64-bitovým. Na rozdiel od procesorov určených pre aplikácie v osobných počítačoch sa dynamicky rozvíja kategória mikroprocesorov so snahou o zlúčenie viacerých funkcií na jeden čip určených pre aplikácie v oblasti riadenia, merania, predspracovania dát a iné. Obvodom, ktoré majú na čipe integrovanú pamäť programu, pamäť dát, hodinový oscilátor, sériový kanál, vstupne/výstupné obvody a mnohé iné funkcie ako A/D, D/A prevodník, Watchdog a i., im prináleží názov mikrokontroléry alebo jednočipové mikropočítače. Na počiatku bol populárny rad 8035 od firmy Intel a neskôr rad 8051. Vďaka vynikajúcim vlastnostiam boli a ešte i teraz sú tieto obvody používané vo veľmi veľkom rozsahu. Mikrokontroléry produkujú mnohí svetoví výrobcovia elektronických súčiastok, pričom niektorí klonovali základný rad INTEL a iní postupovali nezávisle, vzhľadom k pôvodnému vývoju nekompatibilne. Týmto je trh elektronických súčiastok obohatený o mnohé nové druhy mikrokontrolérov rôznych výrobcov. U nás najznámejšími boli mikrokontroléry vytvorené na základe rodiny 8035 a 8051 firmy INTEL.

Ponuka širokej škály veľmi zaujímavých vlastností láka okrem záujemcov o mikroprocesorovú techniku i klasických užívateľov k experimentovaniu, študovaniu novej problematiky, zriaďovanie nových, nákladovo pomerne náročných vývojových prostriedkov a realizáciu svojich zámerov prostredníctvom úplne nových mikrokontrolérov, ktoré im umožňujú to, čo rad 8051 nemohol poskytnúť. Medzi najvýznamnejšie produkty postavené na báze 8051 sú výrobkami firmy ATMEL, ktorá neustále vyvíja mikrokontroléry kompatibilné s radom INTEL. Navzdory tomu, že firmy dodávateľov ponúkajú tieto prvky a v literatúre sa objavili už mnohé aplikačné články, mnohí užívatelia nie sú si vedomí toho, že môžu vziať taký AT89C51, vložiť ho miesto klasického 8051 a on bude bez problémov pracovať. Jedná sa o prvky firmy ATMEL, ktorá má široký sortiment a za veľmi prijateľné ceny. V ponuke sú mikrokontroléry rôzneho výkonu a prevedenia. Zaujímavé sú medzi nimi napríklad málo vývodové typy (dvadsať) pre jednoduchšie aplikácie.

A skutočne veľmi dôležitá je možnosť bez problémov využiť vývojové prostriedky pre rad 8051. Pre tých, čo už pracovali s prvkami rodiny 8051, je to

s ohľadom na skúsenosti značná úspora nákladov. Aplikačných zapojení pre tieto mikrokontroléry je v literatúre mnoho.

V menej náročnom mikropočítačovom riadení a spracovávaní signálov sú v dnešnom štandarde jednočipové 8-bitové mikropočítače od firiem Motorola 68HC05 alebo 68HC11, Intel 8051 a rad jeho variant od rôznych výrobcov (Philips, Siemens, Atmel, Dallas) alebo Zilog Z8, ale i modernizovaný rad Z8CL. S ohľadom na históriu tejto zeme sa najviac u nás rozšírilo použitie procesorov od firmy Intel, ktorá z celosvetového hľadiska nepredstavuje najväčšieho producenta jednočipových procesorov. Procesory od firmy Motorola, ktoré zaujímajú asi tretinu svetovej produkcie, sa u nás presadzujú len pomaly.

Jednočipové mikropočítače dnes nájdeme v mnohých zariadeniach a prístrojoch, s ktorými sa denne bežne stretávame. Mikropočítače riadia kancelárske zariadenia, ako sú digitálne telefóny, faxy, telefónne ústredne, kopírky a tlačiarne. V domácej a spotrebnej elektronike ako sú rádia, televízia, videa, CD prehrávače a zosilňovače, ale i váhy, mikrovlnné rúry a v regulátoroch topenia si ani ich existenciu neuvedomujeme. Každé moderné zariadenie z meracej, automatizačnej a regulačnej techniky si dnes bez mikropočítača ťažko predstaviť. Preto je znalosť návrhu a vývoja jednoúčelových, mikroprocesorom riadených aplikácií v súčasnosti veľmi dôležitá.

1 PREHĽAD O SÚČASNOM STAVE RIEŠENEJ PROBLEMATIKY

Vzhľadom k dynamickému vývoju techniky je zrejmé, že jedným z najperspektívnejších oborov je elektronika a zvlášť tak mikroprocesorová technika. V súčasnej dobe sú k dispozícii mikroprocesory s takými parametrami a cenami, že ich použitie sa stáva potrebným prakticky v každej elektronickej konštrukcií.

Obr. 1 Pohľad na vybavenie triedy pri výučbe mikroprocesorovej techniky

Aplikácie s mikroprocesorom bývajú v porovnaní s klasickým riešením nielen lacnejšie, ale i neporovnateľne "chytrejšie" a elegantnejšie. Bez znalostí mikroprocesorov sa v súčasnej elektronike už pracovať prakticky nedá. Pracovníkov s týmto zameraním (so strednou i VŠ kvalifikáciou) je stále veľký nedostatok. Slovenské firmy sa opakovane presvedčujú, ako je ťažké získať kvalifikovaného zamestnanca. Podobné problémy majú i v zahraničí, kde sa táto situácia rieši dokonca až štátnou podporou imigrácie odborníkov.

Predmet "Mikroprocesorová technika" je atraktívny nielen pre školy s technickým, ale i obecným zameraním (voliteľný/nepovinný predmet alebo záujmový krúžok). Zaistiť kvalitnú výučbu tejto lákavej techniky môže každá škola vybavená počítačmi a to s nízkymi nákladmi a pomerne malými nárokmi na zvýšenie odbornosti pedagógov.

Pre výučbu sú veľmi vhodné jednočipové mikroprocesory ATMEL resp. PIC. Súčiastky i návrhové prostriedky sú výkonné, lacné a práca s nimi sa dá ľahko naučiť. Pre žiadne iné procesory sa nevydáva toľko špecializovaných kníh v slovenčine resp. v češtine.

1.1 PREHĽAD NAJPOUŽÍVANEJŠÍCH RODÍN MIKROKONTROLÉROV

V našich končinách sa ujali najmä mikrokontroléry firiem Intel, Atmel, Microchip, Dallas, Motorola, Philips, Siemens, Zilog... Tie najpoužívanejšie a v praxi veľmi rozšírené podrobnejšie nájdeme v *Prílohe 2, Tab. 1*. Nemožno to brať ako "vyčerpávajúcu studnicu", ale ako základ pre ďalšie štúdium.

1.2 MIKROKONTROLÉRY ATMEL

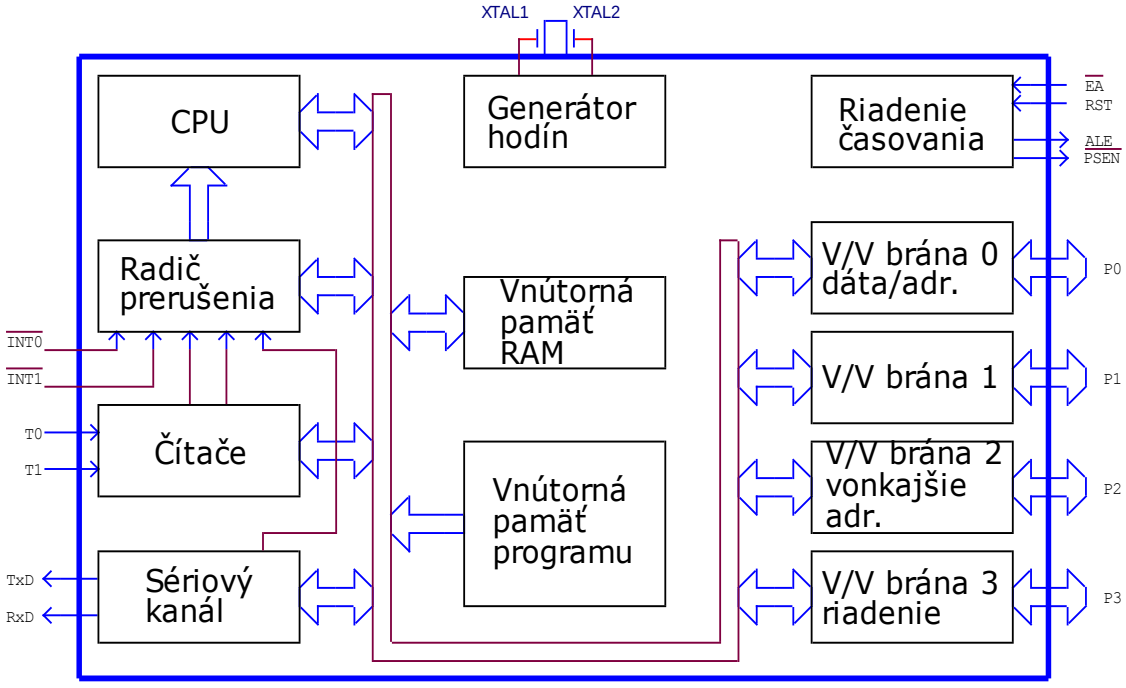
1.2.1 MIKROKONTROLÉRY RODINY 8051

Mikrokontrolér 8051 pochádza z roku 1980 a je vývojovo procesorom relatívne starým. U návrhárov však dosiahol takej obľuby, že i v súčasnej dobe sa veľa výrobcov orientuje na výrobu s jadrom mikrokontroléra 8051, ktoré je rozšírene o ďalšie periférie ako je: pamäť programu o veľkosti 2 kB až 32 kB, ktorú je možné programovať len raz (prevedenie OTP alebo niekoľkokrát (prevedenie EPROM), pamäť EEPROM pre uchovanie konštánt, rozšírená vnútorná pamäť RAM na 256 b, 8 alebo 10-bitový A/D prevodník, obvykle s osem kanálovým analógovým multiplexorom, rozšírené vstupné/výstupné brány, komparačný a záchytný systém alebo dvojlinková prístrojová zbernica I^2C. V súčasnej dobe sú mikrokontroléry ATMEL doplnené ďalej o: pamäť FLASH, rozhranie SPI, Watchdog... Zjednocujúcim základom všetkých týchto rozmanitých typov mikrokontrolérov je vlastné jadro procesoru 8051 (80C51).

1.2.2 JADRO MIKROKONTROLÉRA 8051

Mikrokontrolér 8051 je 8-bitový jednočipový mikropočítač s harvardskou štruktúrou, v ktorej je oddelená programová a dátová pamäť. Mikrokontrolér, ktorého vnútorná štruktúra je blokovo zobrazená na *obr. 2,* je schopný samostatnej činnosti po pripojení vonkajšieho piezo keramického rezonátora (kryštálu) na vývody XTAL1 a XTAL2 a napájacieho napätia 5 V. Na čipe mikrokontroléra je umiestená vlastná procesorová jednotka CPU, ktorá je vnútornou spoločnou zbernicou prepojená s pamäťou programu ROM alebo EPROM o kapacite 4 kB (existujú i typy bez tejto

pamäte), s pamäťou RAM o kapacite 128 bajtov a so štyrmi vstupne/výstupnými bránami P0 až P3, ktoré zaisťujú styk mikrokontroléra s vonkajšími perifériami. Ak nechceme alebo nemôžeme využívať mikrokontrolér v jednočipovej konfigurácií (bez vnútornej ROM/EPROM), sú z čipu vyvedené riadiace signály pre správu vonkajšej programovej (PSEN) alebo dátovej (WR, RD) pamäti, z ktorých každá môže mať až 64 kB. Pre ľahší styk s perifériami je mikrokontrolér vybavený radičom prerušenia, ktorý spracúva 5 zdrojov prerušenia (2 externé, od každého z dvoch časovačov a od sériového kanálu). Jednotlivé prerušenia majú definovanú prioritu na každej z dvoch voliteľných úrovní priority. Čítače, ktoré uľahčujú realizáciu časovania, sú 16-bitové s hodinovým signálom odvodeným z vnútorného generátoru hodín alebo z vonkajšieho vstupu T0 alebo T1. Pre ľahší sériový styk s nadriadenými počítačmi alebo inými spolupracujúcimi mikrokontrolérmi je vybavený duplexným (obojsmerným) sériovým kanálom. Mikrokontrolér je ďalej vybavený Booleovským procesorom, ktorý umožňuje pracovať s jednotlivými bitmi vnútornej pamäti RAM i vnútorných periférií. Pre základnú hodinovú frekvenciu 12 MHz trvajú inštrukcie 1 ms alebo 2 ms s tým, že najdlhšie sú inštrukcie násobenia a delenia, ktoré trvajú 4 ms.



Obr. 2 Vnútorná bloková štruktúra mikrokontroléra 8051

1.2.3 MIKROKONTROLÉRY AT89C A AT89S

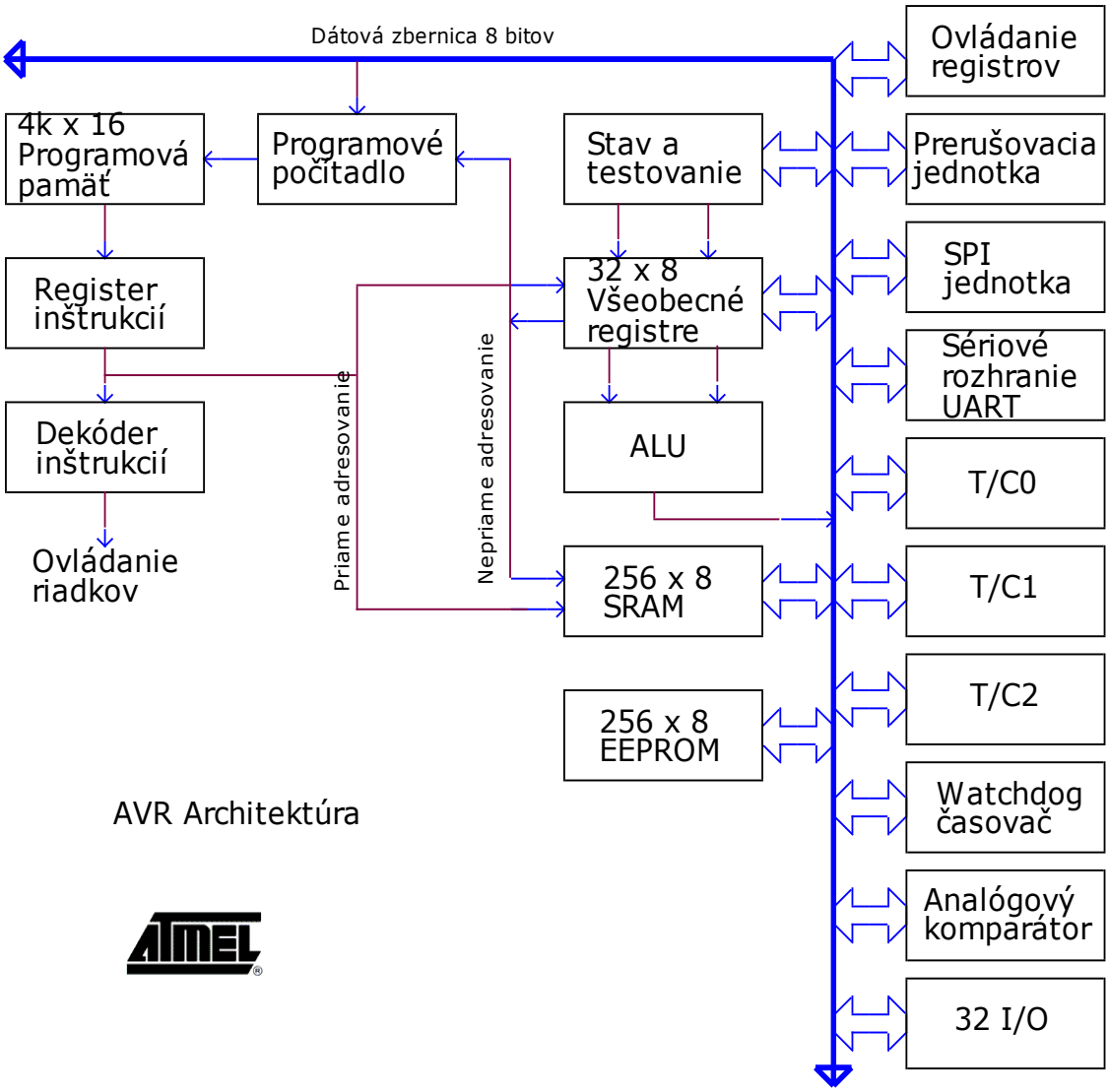
Keď firma ATMEL videla, aký úspech na trhu zaznamenal Intel so svojou rodinou 80C5x, uviedla okolo roku 1993 svoju inováciu tohto obľúbeného mikrokontroléra. Touto inováciou bolo použitie pamäti FLASH ako programovej pamäti namiesto do tej doby používanej EPROM. I keď sa to na prvý pohľad nemusí zdať, bol to veľmi dobrý ťah, lebo do tej doby, aby bolo možné do mikrokontroléra nahrať (preprogramovať) nové programové vybavenie, musel byť zapuzdrený do veľmi drahého keramického puzdra s okienkom. Po odladení programového vybavenia išlo potom pre väčšie série použiť ten istý mikrokontrolér v lacnom plastovom puzdre, ktorý nemal možnosť preprogramovania, lebo puzdro nemohlo mať okienko nutné pre zmazanie informácie uloženej v pamäti EPROM pomocou UV žiarenia. Vďaka veľmi dobre zvládnutej technológii FLASH slávila firma ATMEL veľké úspechy s týmto mikrokontrolérom. To viedlo firmu ATMEL k uvedeniu vlastných "odvodenín" základného typu. Lenže nie všetky aplikácie potrebujú toľko vstupov/výstupov, ako má mikrokontrolér v puzdre DIL40, ako prvá sa objavila na trhu verzia v puzdre DIL20 a to 89C2051 a po nej nasledovali ďalšie ako napr.: 89C1051, 89C1051U a 89C4051.

Pravdepodobným vzorom týchto mikrokontrolérov bol 80C751, ktorý ale nemal implementované niektoré inštrukcie, čo sa ukázalo ako veľká nevýhoda. Nárast zložitosti aplikácií spolu so stále častejším používaním vyšších programovacích jazykov donútili výrobcu implementovať do mikrokontroléra stále väčšiu pamäť programu a v menšej miere i väčšiu pamäť dát. Zložitejšie aplikácie taktiež vyžadovali a vyžadujú nové periférie. Preto sa na trhu objavili ďalšie typy, ktoré tieto potreby uspokojujú. Preto v tomto rade mikrokontrolérov s jadrom 8051 nájdeme obvody s integrovaným obvodom Watchdog, analógovým komparátorom, pamäťou EEPROM pre konfiguračné (kalibračné) dáta, pamäť programu až 32 kB, rozhraním SPI, dvojitým data pointerom. Stručný prehľad týchto mikrokontrolérov je v *Prílohe 2, Tab. 2.*

I keď tieto inovácie spolu so zlepšujúcou sa technológiou priniesli zvýšenie hodinovej frekvencie až na 33 MHz a tým zvýšenie výpočtového výkonu až na 2,75 MIPS, predsa potrebám niektorých aplikácií prestali tieto mikrokontroléry stačiť. Z tohto dôvodu vznikla rodina AVR.

1.2.4 MIKROKONTROLÉRY AVR

Na začiatku 90. rokov minulého storočia sa skupina nórskych návrhárov spolu s programátormi rozhodla navrhnúť novú štruktúru mikrokontrolérov tak, aby štruktúra tohto mikrokontroléra vyhovovala prekladačom vyšších programovacích jazykov, najmä široko používanému jazyku C. Výsledkom snaženia tejto skupiny bolo optimalizované jadro nového radu mikrokontrolérov s harvardskou architektúrou nesúcu hlavné charakteristiky mikrokontrolérov s redukovanou inštrukčnou sadou (RISC − Reduced Instruction Set Controllers). Výsledok môžeme vidieť na *obr. 3* (konkrétne ide o typ AT90S8414). Cez podobnosť s jadrom mikrokontroléra 8051 nájdeme tu podstatné odchýlky. Predovšetkým je to šírka inštrukčného slova, ktoré je 16-bitové. Zväčšenie šírky inštrukčného slova na jednu stranu zväčšilo požiadavky na veľkosť pamäte, na druhú stranu však umožnilo zrýchliť načítanie mnoho inštrukcií, lebo okrem niekoľko výnimiek, vystačia inštrukcie s jedným slovom, t.j. mikrokontrolér ich dokáže načítať behom jedného hodinového cyklu. Druhým viditeľným rozdielom je prepojenie ALU s tzv. poľom 32-pracovných registrov. Táto organizácia ALU spolu s 16-bitovým inštrukčným slovom umožnila návrhárom znížiť počet hodinových taktov potrebných na prevedenie takmer všetkých inštrukcií len na dva takty, načítanie + dekódovanie a vykonanie. Vďaka tomu, že inštrukcie vystačia s jedným slovom, to umožnilo implementáciu jednoduchého prekrývania spomenutých dvoch fáz. Počet hodinových taktov potrebných na vykonanie inštrukcie typu register-register sa týmto znížil na 1 hodinový takt. Ak porovnáme toto so základným inštrukčným cyklom rodiny 80C51, vidíme, že jadro nového radu mikrokontrolérov AVR dokáže poskytnúť 12x vyšší výpočtový výkon pri zhodnom hodinovom taktu.



Obr. 3 Architektúra AVR

Ďalšie "vylepšenie" sa odohralo pri návrhu inštrukčnej sady (vlastných inštrukcií mikrokontroléra, ktoré je schopný vykonávať). Táto časť bola, ako už bolo povedané, riešená v spolupráci s programátormi, tvorcami prekladačov jazyka C. Veľmi dôležitou podmienkou pre generovanie efektívneho kódu pri preklade je existencia data pointerov. Preto boli do štruktúry spočiatku implementované dva tieto pointery (registre), pomenované X a Y. Sama existencia týchto registrov/pointerov nie je postačujúca k efektívnemu prekladu z vyšších programovacích jazykov. Neoddeliteľnou súčasťou je ešte implementácia vhodných módov adresovania.

Analýzou mnoho už napísaných programov bolo ako najvhodnejšie vybrané nasledujúce adresovacie módy:

- nepriame (indirect addressing),
- nepriame s post-inkrementáciou (indirect with post-increment),
- nepriame s pre-dekrementáciou (indirect with pre-decrement),
- nepriame s posunom (indirect with displacement),
- priame stránkové adresovanie (page direct addressing).

Teraz stručne popíšeme jednotlivé spôsoby adresovania. U nepriameho adresovania je hodnota v určenom registri braná ako adresa, na ktorej je uložený operand (hodnota). Rovnako je tomu i u dvoch nasledujúcich adresovacích módov s tým rozdielom, že u prvého dôjde po vyzdvihnutí operanda k zväčšeniu hodnoty v registri, t.j. adresy operanda o 1, u druhého je hodnota v príslušnom registri najskôr o 1 zmenšená a potom použitá ako adresa operanda. Oba spôsoby adresovania sú veľmi vhodné pri práci s premennými dlhšími ako jeden bajt, teda ak už sú to číselné hodnoty, alebo znakové reťazce (strings). Predposledný spôsob adresovania je veľmi vhodný pre adresovanie prvkov štruktúry (príkaz struct v jazyku C) a lokálnych premenných v podprogramoch. I keď inštrukčné slovo má šírku 16 bitov a zdalo sa najprv návrhárom dostatočne dlhé, zostali na hodnote znamenajúcej posunutie iba 4 bity, čo je veľmi málo, lebo skoro vždy je potreba viac. Vzhľadom k horšiemu používaniu priameho stránkového adresovania, zvolili návrhári mikrokontroléra spolu s návrhármi prekladača jazyka C cestu zredukovania, t.j. vynechania, tohto spôsobu adresovania. Tým vznikol väčší priestor pre hodnotu posunutia, ktorá môže byť max. 63, čo vo väčšine prípadov už postačuje (nezabúdajme, na aké úlohy sú tieto mikrokontroléry predurčené).

Vďaka týmto dvom registrom by bolo možné presúvať dáta z jedného miesta (zdroja) na druhé (určené) bez nutnosti manipulácie s hodnotou v pointeroch (ukazovateľoch) ako v prípade iba jedného registra, kde po načítaní jedného bajtu dát musíme zameniť hodnotu v pointeri zo zdrojovej adresy na cieľovú a bajt dát uložiť. Je celkom jasné, že procesor s dvoma pointermi presune dáta omnoho rýchlejšie. Keďže však je nutné vytvoriť v architektúre mikrokontroléra tzv. zásobník, zaberá nám bohužiaľ jeden register práve realizácia tohto zásobníka. Aby návrhári predišli problémom neefektívneho presunu dát s jedným pointerom, pridali do architektúry tretí pointer, síce s obmedzenými vlastnosťami, ale ktorý je pre presun dát plne postačuje. Takže v architektúre mikrokontrolérov nájdeme pointery celkom tri pomenované X, Y a Z.

Pre názornosť si uveďme príklad presunu dát z jedného miesta v pamäti na druhé:

```
      LDI   R16,0x60
Loop:
      LD    R17,Z+
      ST    X+,R17
      SUBI  R16,1
BRNE  Loop
```

Ako je možné i z tohto jednoduchého príkladu vidieť, implementáciu tohto zadania vďaka dostatočnému počtu pointerov a vhodných spôsoboch adresovania tu konkrétne post-increment, prebieha veľmi efektívne.

Vráťme sa teraz k problému vynechaného stránkového adresovania. Tento spôsob adresovania bol nahradený priamym adresovaním s plnou dĺžkou adresy, ktorá je 16-bitová. Tento spôsob adresovania má síce veľkú nevýhodu v tom, že vyžaduje, aby inštrukcia bola dlhá dve slová (dvojslovná), pričom prvé 16-bitové slovo je operačný kód inštrukcie, druhé slovo je 16-bitová adresa, ale získame tým možnosť k prístupu k dátam o veľkosti 64 kB, čo pre mnoho aplikácií postačuje. Zároveň je tento spôsob adresovania vhodný pre implementáciu dátovej triedy static, lebo všetky premenné tejto triedy musia byť umiestené z princípu v pamäti a nie v registroch. I cez uvedenú nevýhodu veľkej dĺžky inštrukcie zostáva tento spôsob adresovania vhodný pre načítanie či úschovu dát typu byte, ako ostatne môžeme vidieť z nasledujúceho príkladu (načítanie hodnoty premennej typu char):

```
Nepriame adresovanie:
      LDI   R30,Low(CharVar)
      LDI   R31,High(CharVar)
      LD    R16,Z
Priame adresovanie:
      LDS   R16,CharVar
```

Z uvedeného je možné vidieť, že v prvom prípade potrebujeme celkom 6 bajtov pamäti programu (tri slová), kde v druhom len 2 slová pamäti programu. Pre prácu s dátami o väčšej šírke, napríklad typu long integer, ktorý je 32-bitový, je vhodnejšie použitie nepriameho adresovania.

```
Nepriame adresovanie:
      LDI   R30,Low(LongVar)
      LDI   R31,High(LongVar)
      LDD   R0,Z
      LDD   R1,Z+1
      LDD   R2,Z+2
      LDD   R3,Z+3
Priame adresovanie:
      LDS   R0,LongVar
      LDS   R1,LongVar+1
      LDS   R2,LongVar+2
      LDS   R3,LongVar+3
```

V prípade prvej varianty je potreba programovej pamäti 12 bajtov, kde v prípade druhom potrebujeme pre rovnakú funkciu 16 bajtov.

Ďalšou oblasťou, kde došlo k veľmi dôležitej optimalizácií inštrukčnej sady mikrokontroléra, sú aritmetické operácie umožňujúce prácu s dátami dlhšími ako 1 bajt. Týmito inštrukciami sú inštrukcie využívajúce príznak Carry a to konkrétne CPC a SBC. Optimalizácia spočíva v odlišnom prístupe k nastavovaniu príznaku Z, t.j. indikácia nulového výsledku. Pre ilustráciu si uveďme nasledujúci vzorový príklad porovnania dvoch 32-bitových čísel. Prvý operand je umiestnený v registroch R3, R2, R1 a R0, druhý operand je v registroch R7, R6, R5 a R4, pričom register R0 a R4 obsahujú najnižšiu časť hodnoty:

```
SUB   R0,R4
      SBC   R1,R5
      SBC   R2,R6
      SBC   R3,R7
BREQ  destination
```

Po vykonaní porovnania v prípade rovnosti oboch operandov by mala inštrukcia BREQ (BRanch if EQual) zaistiť pokračovanie programu na adrese destination. V prípade predchádzajúcich typov mikrokontrolérov však toto nie je pravda, lebo stav príznaku Zero (Z) závisí iba na poslednej inštrukcii, t.j. pokiaľ je najvyšší bajt operandov zhodný, vykoná sa skok na adresu destination bez ohľadu na výsledky porovnania nižších bajtov, čo je chybná interpretácia. Pre korektnú funkciu tohto testu by sme museli pridať za každú inštrukciu SUB či SBI jednu inštrukciu BRNE (BRanch if Not Equal), ktorá by ukončila porovnanie v momente nezhody oboch registrov. Hore uvedené "skomplikovanie", nutnosť pridať tri skokové inštrukcie, vedie u klasickej interpretácie príznaku Zero, jednak k nárastu dĺžky programu, jednak k väčšiemu zaťaženiu mikrokontroléra. Keďže obdobné operácie sú v programoch veľmi časté, pristúpili návrhári mikrokontrolérov k optimalizácií manipulácie s príznakom Zero. Riešenie spočíva v tom, že operácie CPC a SBC môžu príznak Zero iba resetovať (vynulovať), nikdy nie nastaviť. Funkcia, ktorá je realizovaná pre príznak Zero u oboch funkcií, sa dá popísať:

$$Z = (\text{výsledok} == 0)\ \text{AND}\ Z_{old}$$

Výsledný stav príznaku Zero potom závisí na všetkých predchádzajúcich operáciách a nie len na poslednej. Z tohto dôvodu bude hore uvedený príklad na nových mikrokontroléroch AVR pracovať regulárne, ale na štandardných mikrokontroléroch, ako napríklad 89C51, nie. Chovanie príznaku Zero u ostatných operácií je zhodné so štandardnými mikrokontrolérmi.

1.2.5 PREHĽAD RADU AVR

Atmel vyrába mikrokontroléry AVR v troch radách:

Základná

Obsahuje typy: AT90S1200, AT90S2313, AT90S2323, AT90S2343, AT90S4433, AT90S4434, AT90S8515, AT90S8534 a AT90S8535. Tieto typy (s výnimkou AT90S1200) majú 118 inštrukcií a sú u nás dostupné v maloobchodných sieťach. Typ AT90S1200 má 89 inštrukcií a oproti ďalším typom nemá pamäť SRAM. Preto sa k napísaniu SW pre tento typ nedajú použiť niektoré prekladače vyšších jazykov. Na druhej strane výhodou tohto typu je jeho nízka cena.

ATtiny

Obsahuje typy: ATtiny11, ATtiny12, ATtiny15, ATtiny28 a ATtiny26. S výnimkou posledného typu ATtiny26, ktorý má 118 inštrukcií, majú ostatné 90 inštrukcií. Rovnako typy tejto rodiny sú u nás v maloobchode dostupné. Sú to najlacnejšie typy mikrokontrolérov AVR.

ATmega

Obsahuje jednak starší rad: ATmega103, ATmega161, ATmega163 a ATmega323 a novší rad: ATmega8, ATmega16, ATmega64 a ATmega128. Tieto mikrokontroléry majú (s výnimkou ATmega103) 130 inštrukcií. Oproti základnému radu obsahuje navyše inštrukcie pre násobenie. Novší rad je navyše vybavený rozhraním JTAG pre ladenie SW priamo v aplikácii. Väčšina noviniek v oblasti AVR sa týka práve radu ATmega a to ako nových typov mikrokontrolérov, tak i noviniek v oblasti vývojových prostriedkov.

Pri získavaní prvých praktických skúseností s ATMEL AVR MCU pravdepodobne použijeme to, čo je v súčasnej dobe k dispozícii v maloobchodných predajniach, teda MCU základného radu. Ich stručný popis je obsahom *Prílohy 2, Tab. 3 a 4.*

1.2.6 MIKROKONTROLÉR AVR AT90S8535

Vlastnosti:

- AVR® vysoký výkon a nízka spotreba − RISC architektúra:
  - 118 silných inštrukcií − najviac jeden cyklus taktu,
  - 32x8 všestranne použiteľných pracovných registrov,
  - po 8 MIPS priepustnosti pri 8 MHz,
- dátové a permanentné programové pamäte:
  - 8 kB v systémovej programovateľnej pamäti,
  - SPI sériové rozhranie pre systémové programovanie,
  - použiteľnosť: 1 000 cyklov na zápis/zmazanie,
  - 512 bajtov EEPROM,
  - použiteľnosť: 100 000 cyklov na zápis/zmazanie,
  - 512 bajtov vnútornej SRAM pamäti,
  - programovanie zámku pre bezpečnosť prog. vybavenia,
- periférne črty:
  - 8 kanálov, 10 bitov ADC,
  - programovateľný UART,
  - master/slave SPI − Sériové rozhranie,
  - dva 8-bitové časovače/čítače s oddeleným deličom a registrom módov,
  - jeden 16-bitový časovač/čítač s oddeleným deličom, porovnávacími a záchytnými módmi a dvojitý 8, 9, alebo 10-bitový PWM,
  - programovateľný Watchdog časovač s oscilátorom na čipe,
  - analógový komparátor na čipe,
- špeciálne vlastnosti:
  - resetovací obvod,
  - hodiny reálneho času (RTC) s oddeleným oscilátorom a počítadlom módov,
  - externé a vnútorné zdroje prerušenia,
  - tri úsporné módy: Idle, Power Save and Power-down,
- spotreba energie pri 4 MHz, 3 V, 20 °C:
  - aktívny režim: 6,4 mA,
  - idle režim: 1,9 mA,
  - power-down režim: < 1 µA,
- I/O a puzdra:
  - 32-programovateľných I/O liniek,
  - 40-PDIP, 44-PLCC, 44-TQFP, a 44-MLF,
- prevádzkové napätia:
  - VCC: 4,0-6,0 V AT90S8535,
  - VCC: 2,7-6,0 V AT90LS8535,
- rýchlostné stupne:
  - 0-8 MHz pre AT90S8535,
  - 0-4 MHz pre AT90LS8535.

Pinová konfigurácia viď. *Príloha 4*, bloková štruktúra viď. *Príloha 3* a inštrukčný súbor viď. *Príloha 5*.

1.2.7 INŠTRUKČNÝ SÚBOR MIKROKONTROLÉROV S JADROM 8051

Súbor inštrukcií obvodov 8051, teda i mikrokontrolérov ATMEL obsahuje 111 inštrukcií, z toho 49-jednobajtových, 45-dvojbajtových a 17-trojbajtových. Formát inštrukcie sa obvykle skladá z operačného kódu, adresy príjemca a zdroja dát.

Inštrukcie pre prenos dát

Tieto inštrukcie je možné rozdeliť na inštrukcie, ktoré realizujú:

- aritmetické,
- logické,
- s Booleovskými premennými,
- pre prenos dát,
- pre vetvenie programu.

Univerzálne prenosy typu MOV realizujú prenos bitov resp. bajtov (celých osembitových slov) zo zdrojovej adresy na cieľovú adresu. Prenosy s pomocou inštrukcie PUSH prenášajú jeden bajt do zásobníka na adresu, ktorú obsahuje SP register. Podobné prenosy s pomocou inštrukcie POP prenášajú jeden bajt zo zásobníka na cieľovú adresu, pričom sa obsah SP registra po inštrukcii dekrementuje.

Prenosy cez akumulátor využívajú hlavne inštrukcie XCH, XCHD, MOVX a MOVXC. Inštrukcia XCH realizuje výmenu obsahu zdrojovej adresy a akumulátora ACC, inštrukcia XCHD realizuje výmenu nižšej tetrády zdrojovej adresy a ACC. Inštrukcia MOVX prenáša jeden bajt dát medzi vonkajšou pamäťou dát a ACC. Pamäť dát sa adresuje registrom DPTR alebo registrami R0 resp. R1. Inštrukcia MOVC prenáša jeden bajt medzi vonkajšou pamäťou programu a ACC.

Na prenos adresy sa používa inštrukcia MOV DPTR, ktorá naplní register DPTR 16-bitovou konštantou. Žiadna z inštrukcií na prenos dát neovplyvňuje obsah registra PSW.

Aritmetické inštrukcie

Mikrokontroléry ATMEL, kompatibilné s radom obvodu 8051 môžu realizovať všetky štyri základné aritmetické operácie. V aritmetických operáciách možno pracovať

len s 8-bitovými číslami bez znamienka. Pri využití príznaku OV môžu pracovať i s číslami v 2. doplnku.

Medzi inštrukcie na realizáciu aritmetických operácií patrí inštrukcia na dekrementáciu zdrojového operanda (inštrukcia INC), pripočítania zdrojového operanda k ACC (inštrukcie ADD a ACC), inštrukcia na desiatkovú korekciu výsledku (inštrukcia DA). Odčítanie možno realizovať inštrukciami SUBB a dekrementovanie zdrojového operanda inštrukciou DEC.

Inštrukcia MUL realizuje násobenie 8-bitových čísiel bez znamienka, uložených v ACC registri B, pričom výsledok je uložený v ACC (nižší bajt) a v registri B (vyšší bajt).

Inštrukcia DIV realizuje delenie obsahu ACC obsahom registra B, pričom celočíselná časť výsledku je v ACC a desatinná časť v registri B.

Logické inštrukcie

Logické inštrukcie môžu pracovať s jedným alebo dvomi operandami. Pokiaľ inštrukcia pracuje s jedným operandom, je to obsah ACC. Pri inštrukciách s dvomi operandami sa jeden z operandov nachádza v ACC a druhý je v pamäti dát. Logickými inštrukciami s dvoma operandami možno realizovať logické operácie AND (inštrukcia ANL), OR (inštrukcia ORL) a Exclusive OR (inštrukcia XRL).

Inštrukcie na vetvenie programu

V tejto skupine inštrukcií sa nachádzajú podmienené i nepodmienené skoky, ktoré môžu byť krátke v rozmedzí jednej stránky pamäti alebo dlhé v rozmedzí 64 kB. Podobne sa realizujú i dva typy volania podprogramu, volanie podprogramu absolútne (inštrukcia ACALL) a volanie podprogramu v celom rozsahu pamäti programu (inštrukcia LCALL). Mimo toho je možné realizovať nepriamy skok, u ktorého adresa skoku sa určí obsahom registra DPTR, ku ktorému sa pripočíta obsah ACC. Na návrat z podprogramu sa používajú dve inštrukcie: inštrukcia RETI pri návrate z obslužného podprogramu prerušenia a RET pri návratu zo všetkých ostatných typov podprogramov.

Podmienené skoky sú viazané na príznaky Z, NZ, C, NC a hodnoty adresovaných bitov (inštrukcie JB a JNB).

Na vytváraní cyklu máme dva podmienené skoky. Prvý sa realizuje inštrukciou CJNE, ktorá vykonáva porovnanie dvoch operandov a pri nezhode sa realizuje skok.

Druhý sa realizuje inštrukciou DJNZ, ktorá dekrementuje obsah zdrojovej adresy a skok sa realizuje pri jej nenulovom obsahu. V inštrukčnej sade sa tiež nachádzajú inštrukcie pre prácu s portami, s čítačmi/časovačmi resp. pre prácu s prerušeniami.

Prehľad inštrukcií mikrokontrolérov ATMEL je uvedený v *Prílohe 5*.

1.3 MIKROKONTROLÉRY PIC

1.3.1 PREHĽAD VÝVOJA A VLASTNOSTÍ MIKROKONTROLÉROV PIC

Microchip je prvým svetovým výrobcom, ktorý svoje osembitové mikrokontroléry založil na architektúre RISC. Dosiahol tým zaujímavého výsledku: jednak tak vznikli veľmi bohaté, vysoko výkonné osembitové mikrokontroléry (napr. rad PIC18Fxxx), a na druhej strane veľmi rozšírený rad (PIC16C5x, PIC16Cxx, PIC16Fxx a prvé 8-pinové mikrokontroléry na svete PIC12C5xx) jednoduchších, ale elegantných, výkonných a veľmi lacných mikrokontrolérov, ktoré vo svojej kategórií predstavujú najpriaznivejší pomer cena a výkon.

Existuje široká škála jednotlivých typových predstaviteľov, vzájomne sa líšiacich implementovanými technickými prostriedkami, veľkosti EPROM (EEPROM, FLASH) a RAM, počtom I/O pinov, frekvenčným rozsahom, typom oscilátora, puzdrami, teplotným rozsahom a pod. Mnohé vlastnosti, hlavne koncepčné, sú pre všetky typy oboch uvedených rád rovnaké alebo podobné.

Tieto mikrokontroléry sú skutočne jednočipové, nevyžadujú žiadne externé súčiastky. Microchip dôsledne dodržuje zásady RISC − vnútornú obvodovú "jednoduchosť", vysokú ortogonalitu a symetriu. Procesor je harvardský, teda s oddelenými, nerovnako širokými zbernicami a pamäťami pre dáta a pre program.

Program je veľmi úsporný: pamäť programu má optimalizovanú šírku slova (12 až 16 bitov), takže adresa alebo priamy operand (konštanta) je jeho súčasťou. Navyše to predstavuje i výrazné zrýchlenie. ALU má 33 až 58 inštrukcií, všetky o dĺžke jedného slova. Všetky inštrukcie sú jednocyklové, okrem skokových (jednocyklové alebo dvojcyklové, podľa výsledku operácie). Využíva sa dvojstupňový pipelining (fetch, execute), užívateľsky úplne transparentný. Takmer všetky vyhradené registre, príznaky a všetky porty sú uložené do pamäti dát a sú prístupné rovnakými metódami ako užívateľská pamäť dát. Adresovanie je priame, nepriame alebo relatívne.

Inštrukcie môžu "pracovať" priamo v pamäti dát, k dispozícii sú i inštrukcie bitovo orientované. I/O sú obojsmerné (trojstavové), ovládateľné po jednotlivých bitoch. Zásobník má hĺbku 2 až 31 úrovní. Rýchlosť je až 10 MIPS (40 MHz, 100 ns/inštrukcie). Plne statické prevedenie umožňuje i ľubovoľne nízky, prípadne i nulovú taktovaciu frekvenciu.

Technológia je CMOS, napájanie je väčšinou 2,0 až 6,25 V. Vyrábajú sa v teplotných kategóriách: Commercial, Industrial, Automotive.

I/O pinov je 6, 12, 13, 20, 33 alebo 52. Výstupy majú veľkú prúdovú zaťažiteľnosť: 25/20 mA/pin, 40 mA/port. Umožňujú teda i priame budenie LED.

Prúdová spotreba je veľmi nízka. Typické hodnoty sú:

- < 2-5 mA pri 5 V a 4 MHz (podľa typu),
- < 15-100 μA pri 3 V a 32 kHz,
- < 1-3 μA v režime SLEEP pri 3 V a teplote 0 °C až 70 °C.

Základné prevedenie je buď EPROM v keramickom puzdre UV zmazateľné (vhodné pre vývoj), alebo OTP (jedenkrát programovateľné) v plastovom puzdre, pre klasickú i povrchovú montáž. Prakticky všetky nové typy majú pamäť FLASH a dátovú EEPROM.

Režimy činnosti môžu byť:

- HS (high speed) − kryštálový oscilátor, taktovacia frekvencia až 25 MHz, s PLL až 40 MHz,
- XT − kryštálový oscilátor, taktovacia frekvencia do 4 MHz,
- LP, LF (low power, low frequency) − s nízkym odberom, kryštálový oscilátor, taktovacia frekvencia do 40 kHz (typicky 32 768 Hz),
- RC − RC oscilátor namiesto kryštálu (pre minimalizáciu ceny zariadenia),
- IRC − kalibrovaný RC oscilátor integrovaný na čipe (pre maximálne využitie pinov a minimalizáciu ceny zariadenia).

V režimoch HS, XT a LP je možné mikrokontrolér taktovať i externým zdrojom hodinového signálu.

U preprogramovateľných mikrokontrolérov (UV zmazateľných i FLASH) sa režim dá naprogramovať.

Puzdra majú typicky 8, 18, 28, 40, 44, 64, 68, 80 alebo 84 pinov – DIP, SOIC, SSOP, PLCC, PQFP a TQFP.

Všetko je podriadené kritériám:

- rýchlosť, výkonnosť,
- spoľahlivosť a to i bez externých súčiastok,
- nízka cena.

V najjednoduchších mikrokontroléroch (rad PIC16C5x) sú teda implementované iba také obvody, ktoré zabezpečia tieto základné požiadavky. Ostatné funkcie (sériové linky, prerušenia, ...) sa v prípade potreby realizujú programovo.

Knižnice sú k dispozícii. Funkcií realizovaných na čipe je minimum, ale sú navrhnuté tak, aby skutočne, bez úbytku a s rezervou vyhovovali. Sú to hlavne nasledujúce štruktúry:

- power-on reset − je automaticky generovaný vnútornými obvodmi,
- TMR0 − čítač/časovač (8 b) s programovateľným preddeličom (8 b),
- programovateľný Watchdog s vnútorným jedno účelovým RC oscilátorom. Pokiaľ nie je programovo (ale trvale) potlačený, jeho timeout vždy vyvolá reset,
- režim SLEEP so zníženou spotrebou. Iniciuje sa programovo, opustí sa vonkajším signálom reset, vnútorným Watchdog timeout-om, prípadne i inak,
- start-up timer − umožňuje predĺženie resetu po nábehu napájania alebo po "prebudení" z režimu SLEEP (pre spoľahlivý rozbeh kryštálového oscilátora),
- STACK − zásobník s 2-31 úrovňami,
- security fuse - možnosť utajenia naprogramovaného kódu,
- do nevolateľnej pamäti je možné uložiť i užívateľský kód.

Mikrokontroléry PIC sú vhodné prakticky pre všetky embedded (vstavané) aplikácie, pre prácu v reálnom čase, pre flexibilné ovládanie, vyhodnocovanie, konštrukcii periférií, inteligentných dekodérov, ovládačov, ... Vysoká výkonnosť umožňuje ich aplikáciu i v oblastiach, kde ešte nedávno užitie procesora nepripadalo do úvahy ako kvalitatívne vyššia náhrada paralelných zapojení, napr. štandardných obvodov TTL alebo malých hradlových polí.

Okrem samotných technických parametrov sú podstatné i ďalšie aspekty:

- nízke ceny,
- dostupnosť zo skladu i na objednávku, i malé množstvá, krátke dodacie lehoty,
- dostupné, kvalitné, lacné a ľahko zvládnuteľné návrhové a vývojové prostriedky, rozsiahle knižnice funkcií a typických aplikácií,
- je poskytovaná technická a aplikačná podpora.

1.4 MIKROKONTROLÉRY MOTOROLA

1.4.1 VZNIK A VÝVOJ ARCHITEKTÚRY 680X0

Architektúra rodiny mikrokontrolérov Motorola 680x0 bola navrhovaná s veľkým výhľadom do budúcnosti. Už prvý mikrokontrolér 68000 s pracovnou

frekvenciou 8 MHz obsahoval vnútorne celkom 32-bitovú architektúru. Pretože najväčšie dosiahnuteľné puzdro v dobe vzniku mikrokontroléra (rok 1979) bolo puzdro DIL64 a 32-bitová šírka zberníc a pamäti bola ťažko realizovateľná, bola vonkajšia dátová zbernica iba 16-bitová a vyvedených bolo iba 24 bitov adresnej zbernice. Neskôr bola uvedená redukovaná verzia 68008 s 8-bitovou dátovou zbernicou a 20-bitovou adresnou zbernicou a vnútorne rozšírená verzia 68010. Verzie 68HC001 a 68EC000 umožňovali pri resete nastavenie do osembitového alebo šestnásťbitového režimu zbernice. Prvým mikrokontrolérom s 32-bitovou dátovou i adresnou zbernicou bol až mikrokontrolér 68020.

V nedávnej dobe Motorola predstavila novú rodinu mikrokontrolérov − obvody so spoločným názvom NITRON. Pri zavadzaní na trh zvolila firma veľmi zaujímavú stratégiu - po obmedzenú dobu zdarma rozposielala demonštračný KIT, ktorý obsahoval všetko potrebné k zoznámeniu sa s týmito obvodmi (v súčasnej dobe je tento KIT možné zakúpiť za 25$).

1.4.2 RAD NITRON

NITRON je obchodný názov pre mikrokontroléry označené MC68HC908Q… Patrí do rodiny 8-bitových procesorov spoločnosti MOTOROLA (či skôr dnes už Freescale Semiconductor) známej ako HC05. Štruktúra je podriadená zámeru dosiahnuť značného výkonu s výstupmi v puzdre s 8 alebo 14 vývodmi. Podľa toho rozlišujeme označenie T pre 8 vývodov a Y pre 14 vývodov. Procesorové jadro je zhodné s celou rodinou, teda využíva výhody použiteľnosti všetkých vývojových prostriedkov a vyvinutých systémov celej rodiny.

Základné vlastnosti:

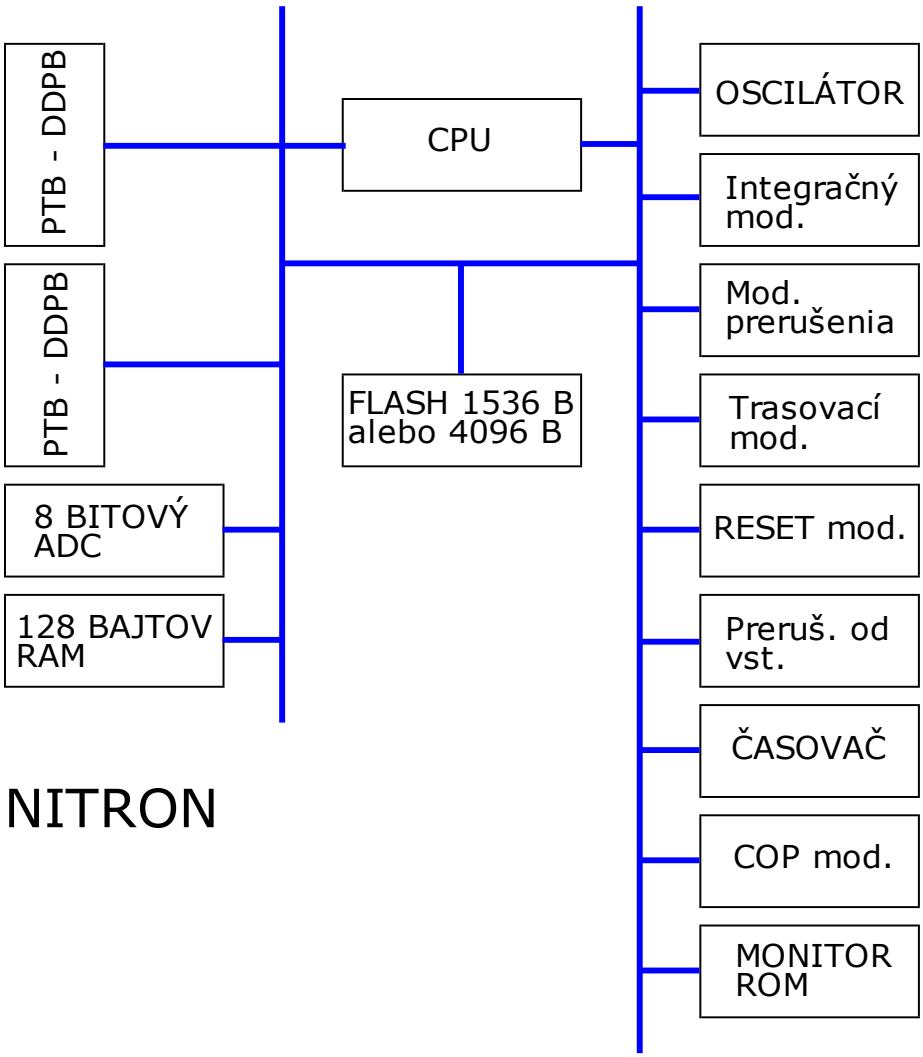
- plne kompatibilné jadro s rodinou M68HC08,
- 5 V alebo 3 V napájacie napätie VDD,
- vnútorná zbernica na 8 MHz pre napätie 5 V, 4 MHz pre napätie 3 V,
- možnosť nastavenia vnútorného oscilátora − vlastný oscilátor na 3,2 MHz je možné nastaviť v 8-bitovom móde. Možnosť nastavenia ± 25 % a v nastavenom režime je odchýlka ± 5 %,
- vlastné programovanie FLASH pamäti spolu so zabezpečením dát,
- 128 bajtov RAM pamäti,
- kanály 16-bitového časovača,
- 6-bitový systém prerušenia od vstupov,
- softvérové nastavenie konfigurácie obvodu vrátane Watchdog-u.

Tab 1. Parametrické rozlíšenie obvodov

Typ	Flash	ADC	I/O	Puzdro
MC68HC908QT1	1536 B	-	5	8 pin PDIP, SOIC
MC68HC908QT2	1536 B	4 kanály 8 bit	5	8 pin PDIP, SOIC
MC68HC908QT4	4096 B	4 kanály 8 bit	5	8 pin PDIP, SOIC
MC68HC908QY1	1536 B	-	13	16 pin PDIP, SOIC, TSSOP
MC68HC908QY2	1536 B	4 kanály 8 bit	13	16 pin PDIP, SOIC, TSSOP
MC68HC908QY4	4096 B	4 kanály 8 bit	13	16 pin PDIP, SOIC, TSSOP

Blokové usporiadanie

Blokové usporiadanie najdôležitejších častí je na nasledujúcej schéme. S okolitým prostredím komunikuje cez brány PTA, prípadne PTB. Ostatné bloky sú prístupné, alebo sú parametricky dosiahnuteľné cez komunikačné protokoly.



Obr. 4 Blokové usporiadanie radu NITRON

Komunikácia

HW prostriedkov zaisťujúcich komunikáciu s vývojovým prostredím je celý rad a líšia sa komfortom a samozrejme cenou. Pre malý systém je ale potreba malý a hlavne dostupný systém. Tento predstavila česká sekcia MOTOROLA pod názvom JANUS. Presný popis a stavebný návod nájdu záujemcovia v (The small Word of HC08). To isté platí o SW − jednoducho dostupný. Pre NITRON prichádza do úvahy vývojové prostriedky pre ASM alebo C. Opäť je možno ich zohnať ako voľne šíriteľné verzie na predchádzajúcej web adrese.

1.5 HARDVÉR PRE MIKROKONTROLÉRY ATMEL

Pred desiatimi, resp. pätnástimi rokmi vývojári v oblasti mikroprocesorovej techniky mali značné problémy pri osadzovaní mikroprocesorov v praxi. Mnoho z nich zháňalo základné softvérové vybavenie, t.j. prekladač zdrojového kódu, či relokátor (INTEL, Avocet). V oblasti hardvérového prostredia bola situácia ešte komplikovanejšia, pretože len málo pracovísk bolo vybavené komerčnými vývojovými prostriedkami. Nedostatok financií sa obchádzal budovaním vlastných pomôcok, ku príkladu simulátory pamäti a programátorov EPROM.

V súčasnej dobe je situácia odlišná v niekoľkých smeroch. Zaobstaranie editorov, prekladačov, relokátorov, softvérových simulátorov je často bezplatné. Jednak výrobcovia mikroprocesorov a mikrokontrolérov ponúkajú lacné získanie "Start Kit-ov", v aplikačnej oblasti je veľa subjektov ponúkajúcich rôzne riešenia vývojových a konštrukčných pomôcok, až sa niekedy zdá, že je ich viac ako užívateľov. Stačí si "zasurfovať" po internete a nevieme, čo si vybrať.

Ak už používame akékoľvek hardvérové prostriedky, pri prechode na mikrokontroléry ATMEL nevzniknú žiadne podstatné prekážky. V prípade použitia mikrokontrolérov AT... 53, 55, 252 môže byť nepríjemné, že niektoré SFR prekladač nepozná a musíme preto použiť priamu adresáciu, alebo ju deklarovať priradením EQU.

1.5.1 LPT ISP PROG

Interface (rozhranie) medzi paralelným portom a ISP rozhraním

Programovanie mikrokontrolérov ATMEL pomocou rozhrania ISP je dnes už bežná vec. Šetrí to čas pri vývoji, nie je treba stále prenášať CPU medzi programátorom a päticou v zaradení. Chceme Vám predstaviť jednoduchý programátor, alebo skôr interface (rozhranie), ktoré vytvára ISP (In System Programming) pomocou LPT portu.

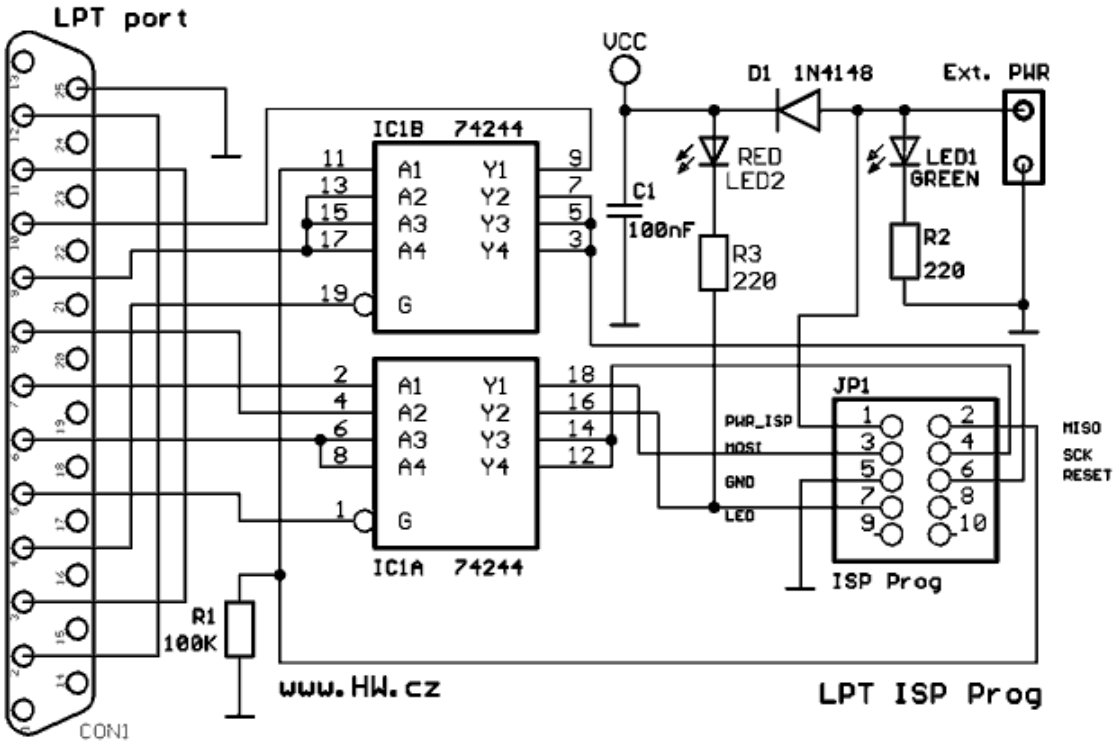
LPT ISP Prog je skôr interface (rozhranie), než programátor v pravom slova zmysle. Neobsahuje žiadne pätice na programovanie obvodov priamo na základnej doske. Jeho výstupom je iba konektor ISP. Pokiaľ potrebujeme naprogramovať ISP procesor mimo aplikácie, je možné samozrejme pripojiť ISP priamo na procesor, doplniť k procesoru kryštál s kondenzátormi a je to. Spôsob realizácie dokumentuje (Řehák, Jan. LPT ISP Prog).

Výhody programátora:

- pripojenie na paralelný port – neblokuje sériový port. Ten tak zostáva použiteľný pre pripojenie ladeného zariadenia,
- softvér pre Windows 95, 98 i Windows NT/2000/XP (SW obsahuje rýchlejšie drivery (ovládače), ktoré fungujú pod W9x, alebo pomalšie i pre NT a W2000/WXP),
- na rozdiel od prameho pripojenia pinu paralelného portu obsahuje tento programátor budič zbernice, ktorý definuje logické úrovne výstupu, takže programátor funguje na všetkých paralelných portoch korektne nemusí byť pripojený extrémne krátkym káblom,
- vďaka použitiu budiča, sú hrany SCK dostatočne strmé a nehrozia problémy popisované v Errata dokumentoch ATMEL-u (pokiaľ samozrejme nepoužijeme 10 metrov dlhý kábel),
- softvér pre LPT ISP Prog je súčasťou štandardného balíka "PonyProg" (LANCONELLI, C. PonyProg serial device programmer), alebo CodeVisionAVR. Vďaka tomu je softvér priebežne inovovaný a aktualizovaný. Dá sa teda počítať s vyriešením problému, pokiaľ napríklad ATMEL upraví programovacie algoritmy obvodov,
- obslužný SW je veľmi príjemný, je napísaný pre priebežný vývoj. Obsahuje napríklad voľbu, ktorú je možné pri každom programovaní .hex súboru načítať znovu z disku, čo je nutné pri akomkoľvek vývoji a kompilácií kódu v inom okne,
- programátor obsahuje signalizáciu priebehu programovania a signalizáciu napájania.

Prevedenie LPT ISP Prog

Vzhľadom k histórii bola konštrukcia navrhnutá na dvoch plošných spojoch. Jednotlivé verzie sa schematicky líšia iba v konektoroch pre zapojenia výstupov.



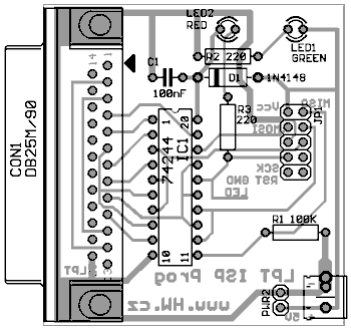
Obr. 5 Schéma zapojenia programátora LPT ISP Prog. (HW Server)

Napájanie

ISP rozhranie obsahuje napájacie napätie. Pretože tu existuje veľa možností napájania, je na doske programátora osadená dióda D1. Pokiaľ ju skratujeme, je všetko napájanie pripojené. Dióda oddeľuje iba napájanie pre samotný oddeľovací obvod, napájacie piny z ISP konektora sú pripojené so svorkovnicou pre prípadné externé napájanie.

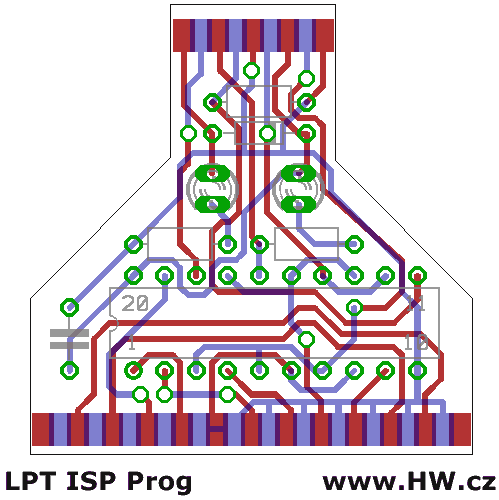
ISP okrem napájania obsahuje ovládanie resetovania mikrokontroléra (/RESET), vodiča pre sériový zápis (MOSI) a čítanie (MISO) obsahu interných pamätí FLASH a EEPROM + jeden vodič pre synchronizáciu prenosu dát (SCK). Vodiče MISO a MOSI nie sú krížené t.j. MISO z programátora sa pripojuje na MISO CPU a MOSI na MOSI.

Verzia na jednostranný plošný spoj



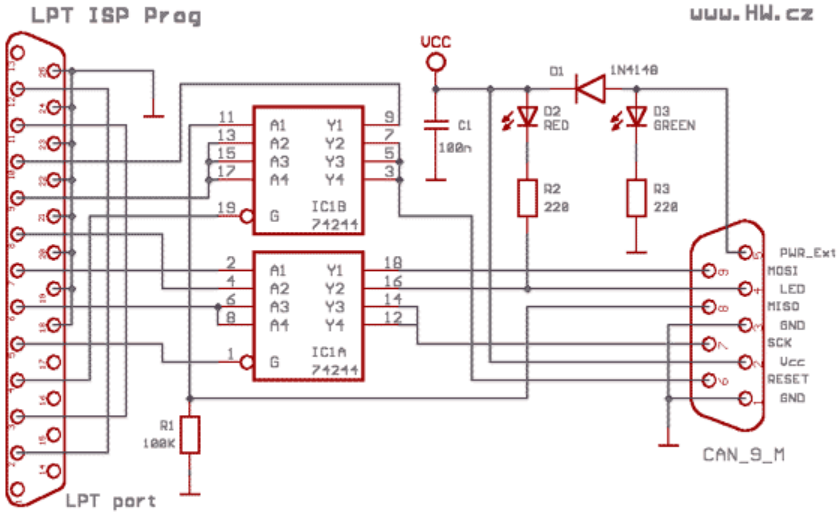
Obr. 6 Plošný spoj programátora LPT ISP Prog. (HW Server)

Verzia na plošný spoj do krytu redukcie CANNON 25/9



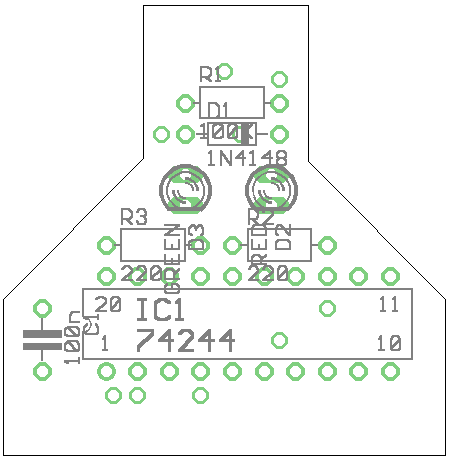
Obr. 7 Plošný spoj programátora LPT ISP Prog − redukcia. (HW Server)

Schéma verzie programátora LPT ISP v redukcií



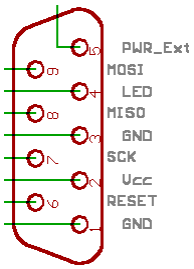
Obr. 8 Schéma programátora LPS ISP Prog − redukcia. (HW Server)

Osadenie plošného spoja programátora v redukcii



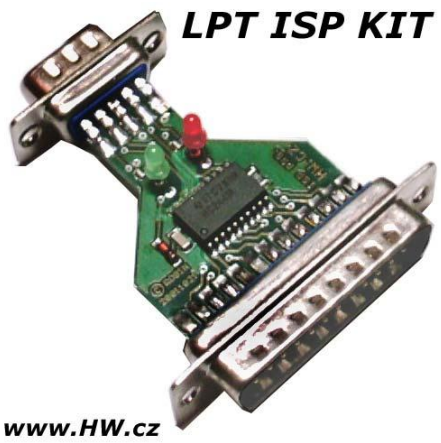
Obr. 9 Osadenie DPS pre LPT ISP Prog − redukcia. (HW Server)

Detail zapojenia výstupného konektoru verzie v CANNON redukcii



Obr. 10 Detail zapojenia konektoru Cannon pre programátor LPT ISP - redukcia. (HW Server)

Zobrazenie osadenej DPS



Obr. 11 Osadenie DPS programátora LPT ISP Prog − redukcia. (HW Server)

Celkový pohľad na programátor v redukcii

Obr. 12 Celkový pohľad na programátor LPT ISP Prog − redukcia. (HW Server)

Porovnanie s AVR ISP Prog

Programátor AVR ISP Prog, ktorý je zapojený podľa Aplication Note 910 priamo z ATMEL-u. Časom však prax priniesla niekoľko problémov:

- AVR ISP Prog sa pripojuje k RS232, ktorý však často potrebujeme pre konkrétnu aplikáciu, a pokiaľ nemáme voľné dva sériové porty, prináša to problémy,
- softvér pre AVR ISP Prog je veľmi nepríjemný pre priebežnú prácu,
- originálny SW od ATMEL-u nepracuje správne s procesormi 89C8252.

1.5.2 START KIT-Y PRE AVR

Start Kit-y predstavujú kompletný balík vývojových nástrojov pre mikrokontroléry Atmel AVR a s mikroprocesormi vo všeobecnosti vôbec. Dáva užívateľom slobodu pri vývoji, ladení a testovaní aplikácií a prototypov.

Veľmi obľúbené sú výrobky od firmy Kanda, ktoré sú dostupné i u nás.

1.5.3 STK 500

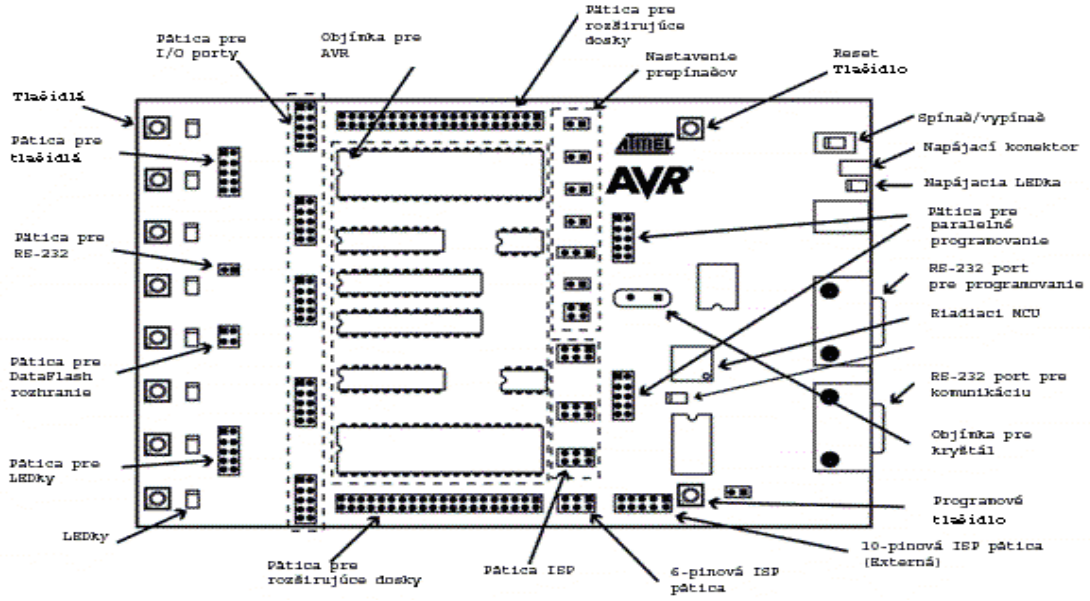
AVR STK500 Start Kit podporuje všetky druhy programovania mikrokontrolérov AVR ukladaných do pätíc rovnako ako ISP programovanie v externých zariadeniach. Bližší popis STK 500 obsahuje dokument (KYLE, J. P. STK500 Protocol AVR Bootloader).

Základné vlastnosti:

- spolupracuje s AVR Štúdiom,
- rozhranie RS232 pre pripojenie k PC,
- regulovaný napájací zdroj pre vstupné napätie 10-15 Vst,
- sokly pre 8, 20, 28 a 40-vývodové mikrokontroléry AVR,
- paralelné a sériové programovanie vysokým napätím,
- sériové ISP programovanie,
- ISP programovanie v externých aplikáciách,
- preprogramovanie z AVR zariadení,
- 8 tlačidiel pre použitie v užívateľskej aplikácii,
- 8 LED pre použitie v užívateľskej aplikácii,
- všetky vývody AVR sú ľahko prístupné cez vyvede40né konektory,
- druhý RS232 port pre použitie v užívateľskej aplikácii,
- rozširujúce konektory pre Plug-in moduly a prototypy,
- Mbit dátovej FLASH,
- frekvencia oscilátora, napätie a nulovanie je nastaviteľné i z AVR Štúdia.

Obr. 13 AVR STK500. (Atmel Corporation homepage)

Vstupné/výstupné porty (I/O) sú užívateľovi sprístupnené prostredníctvom konektorov, takže je ich možné použiť v spojení s externými zariadeniami, alebo je možné využiť ovládacích tlačidiel a signalizačných LED umiestených priamo na vývojovej doske. Samostatný sériový port RS232 môže byť pripojený k akýmkoľvek I/O vývodom.



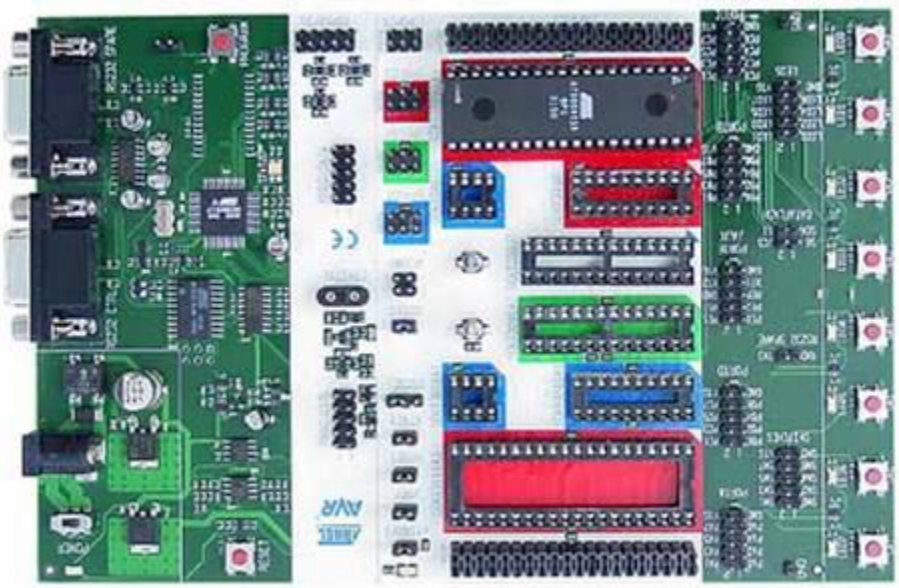
Obr. 14 Rozmiestenie konektorov na vývojovej doske AVR STK500

Pre potreby vývoja aplikácií s rôznymi mikrokontrolérmi je vývojová doska STK500 koncipovaná ako široko variabilná, takže je možné nastavovať prakticky všetko od frekvencie oscilátora až po napájacie napätie mikrokontrolérov pomocou skratovacích prepojok. K dispozícii je tiež objímka pre pripojenie externého kryštálu.

Pri použití vnútorného oscilátora dosky však je možné hodinovú frekvenciu mikrokontrolérov AVR a ich napájacie napätie rovnako pohodlne nastavovať z prostredia AVR Štúdia.

Tab 2. Podporované mikrokontroléry

AT89	**AT90**	**ATtiny**	**ATmega**
AT89951 [3]	AT90S1200	ATtiny11	ATmega8
	AT90S2313	ATtiny12	ATmega8515
AT89952 [3]	AT90S2323	ATtiny15	ATmega8535
	AT90S2333	ATtiny22	ATmega161
	AT90S2343	ATtiny26	ATmega162
	AT90S4414	ATtiny28	ATmega163
	AT90S4433		ATmega16
	AT90S4434		ATmega169 [2]
	AT90S8515		ATmega323
			ATmega32
	AT90S8535		ATmega64 [1]
			ATmega103 [1]
			ATmega128 [1]

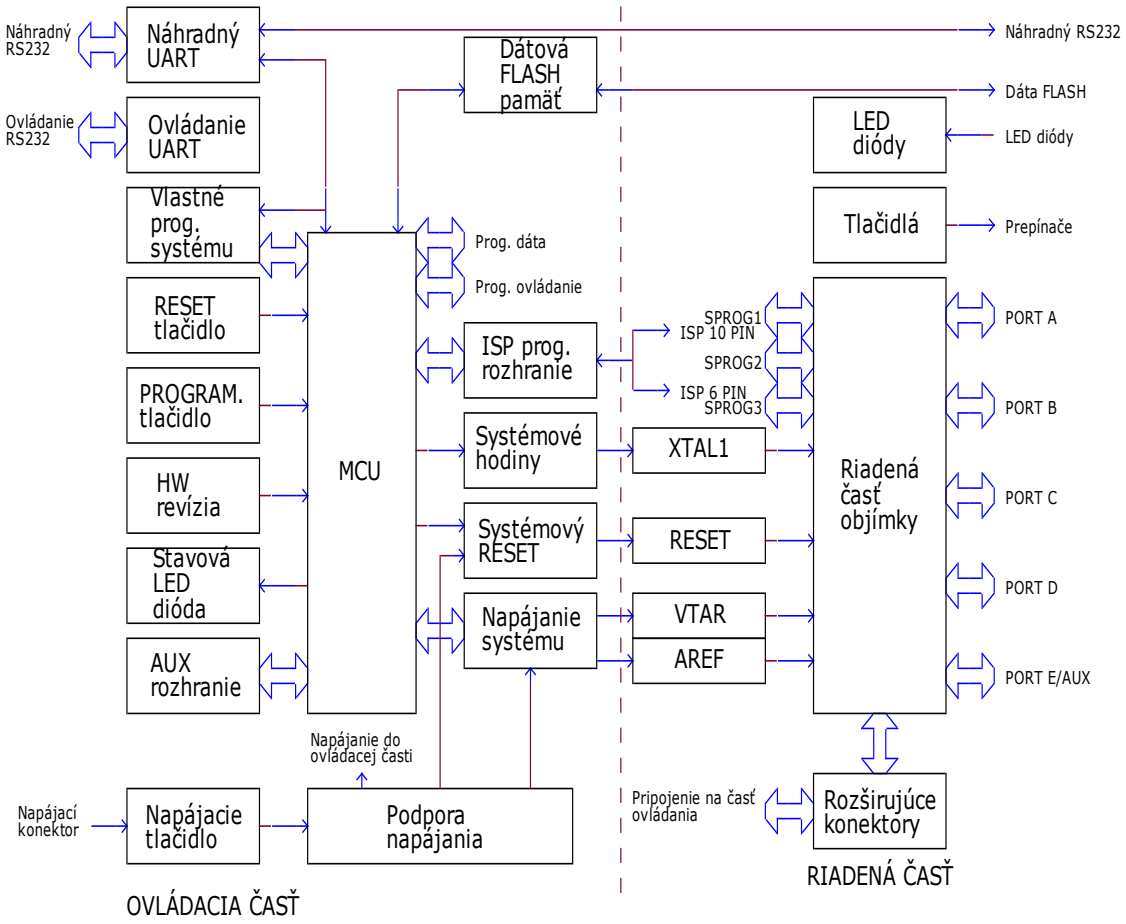


Obr. 15 Pohľad na vývojový Start Kit AVR STK500. (Atmel Corporation homepage)

Poznámky:

- podporovanie pri ISP programovaní v externej aplikácii alebo s rozširujúcim modulom STK501,
- podporovanie pri ISP programovaní v externej aplikácii alebo s rozširujúcim modulom STK502,
- podporovanie pri ISP programovaní v externej aplikácii,
- podporované sú rovnako nízkonapäťové prevedenia.

Programovacie rozhranie pre AVR STK500 je integrované vo vývojovom prostredí AVR Studio dodávanom spolu so Start Kit-om. Pamäti FLASH a EEPROM môžu byť programované samostatne rovnako ako ich poistky (Fuses) a zámky (Lock bit), ktoré je možné nastavovať v rámci programovania pamätí, či neskôr.



Obr. 16 Bloková schéma vývojovej dosky AVR STK500

1.6 SOFTVÉROVÉ PROSTRIEDKY A LADENIE PROGRAMOV

1.6.1 PROGRAMOVACIE TECHNIKY

Dnešné mikrokontroléry sú tak komplexne hardvérovo vybavené, že nezriedka vznikajú aplikácie s minimálnym dodatočným obvodovým doplnkom a preto sa celková inteligencia a výkonnosť preniesla do softvéru.

Pri realizácií programu je nutné poznať architektúru mikrokontroléra, možnosti jeho inštrukčnej sady, ale i hardvérovo-softvérového vybavenia, na ktorom budeme aplikáciu vyvíjať. Nutným predpokladom je však dôkladná znalosť požiadaviek kladených na riešenie.

Tak sa postup v zásade dá popísať v niekoľkých bodoch:

- dôsledná analýza požadovanej aplikácie, návrh riadiacich a signálových rozhraní, návrh algoritmu, prieskum knižných modulov, pokiaľ to ide i štúdium konkurenčných riešení alebo vlastností,
- realizácia spoľahlivého skúšobného zapojenia aplikácie, v prípade rozsiahlejších aplikácií, po častiach. Pomoc skúseného obvodára je k nezaplateniu,
- zapísanie logickej kostry programu, rozdelenie do menších celkov, realizácia podprogramu a programových modulov,
- vybrať vhodný programovací jazyk, pokiaľ je to možné:
    - assembler − jazyk symbolických adries je vhodný predovšetkým pre vývoj aplikácií vyžadujúcich rýchlosť pri menších nárokoch na pamäť programu i dát,
    - vyššie programovacie jazyky (C, C++, Basic, Java), už umožňujú rýchlejší vývoj aplikácií a prehľadný vývoj náročnejších aplikácií,
- ladenie pozostáva v zásade z opakujúcich sa prekladov zdrojových textov do výkonného modulu v strojovom tvare, zavádzanie do emulačnej pamäti, spustenie v aplikáciách a sledovanie nadobúdajúcich vlastností aplikácie,
- ak je aplikačný program odladený a aplikácia vykonáva činnosť podľa zadania, je preložený hotový program v strojovom tvare (formát .HEX, .IH, .BIN) zavedený do pamäti programu, v našom prípade vnútornej FLASH EPROM a mikrokontrolér je definitívne umiestený do aplikácie,
- posledným krokom, nie však významom, je finalizácia konštrukčného riešenia celej aplikácie.

1.6.2 VÝBER PROGRAMOVACIEHO JAZYKA

K vývoju aplikácií pre jednočipové mikropočítače je možné použiť celý rad programovacích jazykov (jazyk symbolických inštrukcií − Assembler, C, Pascal, Java, Basic, atď.). Dominantné postavenie si stále udržuje jazyk symbolických inštrukcií s viac než 50 %. Súčasný trend však stále viac smeruje k použitiu vyšších programovacích jazykov, menovite jazyka C.

Pre mnohých vývojárov sa však jedná o zásadný problém, ktorý je možné nazvať strachom z niečoho nového, zložitého a neznámeho. Prečo prechádzať od zavedeného programovania v jazyku symbolických inštrukcií k niečomu inému? Na túto otázku a ďalšie by mal odpovedať nasledujúci text.

Pohľad na vývoj z hľadiska aplikácie

Komplexnosť súčasných jednočipových aplikácií z oblasti automobilového priemyslu, GSM a ďalších oblastí, spolu s narastajúcim tlakom na spracovanie aplikácií jasne smerujú k použitiu vyšších programovacích jazykov.

Výhody použitia vyšších programovacích jazykov pri spracovávaní aplikácií:

- zníženie doby vývoja a tým spojených nákladov,
- sprehľadnenie zdrojových kódov,
- dlhodobá správa a údržba projektu po garantovanú dobu podpory pre dané zariadenie,
- dôraz na multiplatformnosť.

Výrobcovia mikrokontrolérov tento trend podporujú a stále častejšie integrujú väčšie množstvo pamäti RAM a FLASH priamo na čipe. V základnom popise týchto mikrokontrolérov je priamo vyzdvihované použitie v spojitosti s vyššími programovacími jazykmi.

Prečo použiť jazyk C?

Jazyk C bol od prvopočiatku navrhovaný k systémovému (nízko úrovňovému) programovaniu. Vďaka tomu sa presadil všade tam, kde je potreba priamo ovládať ľubovoľný hardvér. V jazyku C je napísaných mnoho operačných systémov, užívateľských a jednočipových aplikácií. Prekladače jazyka C existujú pre väčšinu dostupných typov procesorov.

Výhody jazyka C:

- jednoduchá prenositeľnosť zdrojových kódov programu (multiplatformnosť),
- prehľadnosť zdrojových kódov programu a celkové zjednodušenie pri správe zložitých projektov,
- rýchlejší vývoj aplikácií (použitie štandardných knižníc),
- nástroje na optimalizáciu a validáciu výsledného kódu programu,
- zavedený programovací jazyk, podporovaný výrobcami hardvéru (mikroprocesorov) a softvéru (kompilátorov).

Nevýhody jazyka C:

- vyššia cena kvalitného vývojového prostredia,
- relatívne väčšie nároky na pamäť dát a programu výslednej aplikácie,
- zložitejšie na osvojenie.

Implementácia jazyka C u mikrokontrolérov x51

Vďaka neutíchajúcej obľube mikrokontrolérov x51 je k dispozícii dostatočné množstvo kompilátorov jazyka C. Základné vlastností jazyka C je, že využíva intenzívne prácu so zásobníkom (predávanie parametrov, lokálne premenné, návratové adresy funkcií, reetrantné funkcie, atď.). To je kameňom úrazu u mikrokontrolérov x51, ktorá nie je obdarená vhodným zásobníkovým systémom. Úspešnosť implementácie jazyka C u mikrokontrolérov radu x51 závisí na tom, ako sa daný výrobca kompilátoru stotožnil s týmto neľahkým problémom. V ďalšej fáze sa potom riešia špecifické problémy s implementáciou jazyka C na 8-bitový mikrokontrolér radu x51:

- prístup k vnútorným a externým perifériám,
- správa prerušenia,
- optimálne využitie obmedzenej inštrukčnej sady,
- špecifické vlastnosti rozdielnych pamäťových priestorov,
- podpora rôznej konfigurácie pamätí ROM a RAM,
- vysoká úroveň optimalizácie pre maximálne využitie kódového priestoru,
- prepínanie registrových bánk,
- podpora rôznych klonov štandardných s 8051.

Počet kompilátorov jazyka C pre mikrokontroléry radu x51, ktoré môžeme nájsť na internete sa ľahko vyšplhá k číslu dvadsať. Ako sa v takom počte kompilátorov jednoducho zorientovať a podľa čoho vybrať ten správny?

Prvá vec, čo by mal každý záujemca urobiť, je navštíviť internetové stránky daného výrobcu. U väčšiny výrobcov je možné voľne získať ukážkovú verziu ich produktu k vyskúšaniu.

Tieto voľné verzie sú rôzne limitované:

- veľkosťou výsledného kódu aplikácie,
- počiatočnou adresou, na ktorej je aplikácia zostavená,
- časovým obmedzením doby používania,
- absenciou niektorých knižníc (napr. práca s pohyblivou čiarkou),
- dostupnými pamäťovými modelmi,
- počtom podporovaných klonov mikrokontrolérov x51.

Takto voľne získaná verzia programu, často plne uspokojí potreby začiatočníka, študenta, či amatéra. S dostupnými aj keď limitovanými nástrojmi je možné vytvoriť mnoho funkčných a zaujímavých aplikácií, špeciálne vhodných pre základný rad mikrokontrolérov firmy Atmel, veľmi obľúbených u tejto skupiny vývojárov.

V oblasti GNU kompilátorov pre mikrokontroléry radu x51 je k dispozícií iba jeden zástupca s názvom SDCC (Small Device C Compiller). Svojimi vlastnosťami výsledného kódu programu sa riadi do strednej triedy kompilátorov. Pri realizácií zložitejších projektov je však potreba počítať so zvýšeným úsilím zo strany vývojára k dosiahnutiu patričného výsledku. Pokiaľ sa budeme zaujímať vývojom aplikácií na profesionálnej úrovni nezostane nám nič, než sa poobzerať po komerčnom riešení.

1.6.3 CHARAKTERISTIKA NÁSTROJOV PRE VÝVOJ APLIKÁCIÍ V JAZYKU C

Pokiaľ budeme investovať nemalé peniaze do nákupu vývojového prostredia, mali by sme si uvedomiť čo od daného produktu chceme a k čomu ho budeme používať. Článok porovnáva jednotlivé existujúce C kompilátory na x51 a upozorňuje na prehliadané ale dôležité vlastnosti jednotlivých kompilátorov.

Pri výbere sa treba zamerať na nasledujúce body:

- integrované vývojové prostredie (IDE),
- správa projektu a preklad (inteligentný make),
- podporované klony x51 (sú využívané nové inštrukcie),
- softvérový debugger (simulácia nových periférií u klonov x51),
- naviazanosť na ďalšie externé programy (CVS, PC-Lint, programovací SW, …),
- výstupný formát kompilátora (BIN, Intel-HEX, OMF-51, …),
- optimalizácia kódu aplikácie (je možné vytvoriť aplikáciu > 64 kB),
- pamäťové modely (vnútorná/externá CODE a XRAM),
- náchylnosť k chybám pri preklade,
- podpora štandardu ANSI C (do akej miery je kompilátor schopný preložiť zdrojový kód v C),
- rozsah implementovaných štandardných funkcií jazyka C,
- pokračuje výrobca v ďalšom vývoji kompilátora, updaty, užívateľská podpora?
- dostupná dokumentácia k danému kompilátoru.

Zoznam kompilátorov jazyka C a najbežnejšie používané IDE sú bližšie popísané v *Prílohe 6*.

2 CIEĽ PRÁCE

Cieľom diplomovej práce je vytvoriť ucelené vývojové pracovisko s RISC mikrokontrolérmi, ktoré by umožnilo rýchly a efektívny vývoj, testovanie a odladťovanie rozličných aplikácií v pedagogickom procese a následné nasadenie v praxi.

Vývojové pracovisko má spĺňať:

- jednoduchosť konštrukcie,
- finančnú nenáročnosť,
- rýchly vývoj a testovanie,
- pedagogické využitie.

Vývojový nástroj je osadený jednočipovým mikrokontrolérom AT90S8535 so základnými perifériami. Programovanie vnútornej pamäti FLASH a EEPROM je riešené ISP programovaním.

3 METODIKA PRÁCE

Charakteristika prostredia

Predmetom nášho úsilia bude skonštruovať experimentálne vývojové pracovisko, ktoré svojím charakterom bude slúžiť ako pomôcka v predmete zaoberajúcom sa oblasťou mikroprocesorovej techniky.

Konečný výsledok našej práce by mal vychádzať z komerčných Start Kit-ov, ktoré vyvíjajú renomované firmy, ktoré sú dlhoročnými výrobcami procesorov vo všeobecnosti vôbec. Sú to napr. firmy: ATMEL, INTEL, MICROCHIP, MOTOROLA, DALLAS a pod.

Opis experimentálneho pracoviska

Navrhované vývojové pracovisko bude obsahovať tieto funkčné bloky:

- programovacie a odladťovacie rozhranie,
- napájanie,
- výstupná signalizácia:
  - LED,
  - 7-segmentový displej,
  - LCD,
  - popr. zvuková,
- snímanie vstupov od užívateľa:
  - tlačidlá,
  - maticová klávesnica,
- ovládanie externých zariadení pomocou spínacích prvkov,
- využitie prerušovacieho systému,
- realizácia A/D, D/A prevodníka pre účel merania,
- komunikácia s PC,
- pripojenie externej pamäti,
- realizácia zberníc: I^2C, MicroWire, 1-Wire, SPI.

Opis zdrojových príkladov experimentálneho pracoviska:

- ovládanie LED diód − bude riešiť jednoduché zapojenie výstupnej signalizácie v podobe 8 LED diód k mikrokontroléru. Obsah jednotlivých príkladov bude nasledujúci:
  - program č. 1: Blikanie 8 LED diód (párne/nepárne), pripojených k portu A, pomocou 2 premenných. Frekvencia blikania bude 2 Hz,
  - program č. 2: Postupné zapínanie 8 LED diód (0. ==> 7.) na porte A, pomocou 8 premenných. Frekvencia blikania bude 2 Hz,
  - program č. 3: Postupné zapínanie 8 LED diód (0. ==> 7.) na porte A, pomocou 8 rozmerného poľa. Frekvencia blikania bude 2 Hz,
  - program č. 4: "Night Rider" z 8 LED diód na porte A, pomocou 8 rozmerného poľa. Frekvencia blikania bude 20 Hz,
- komunikácia s LCD − bude pojednávať o pripojení LCD k mikrokontroléru. Obsah jednotlivých príkladov bude nasledujúci:
  - program č. 5: Vyslanie reťazca "Ahoj" na LCD displej pripojený na PORTB,
  - program č. 6: LCD − užívateľom definovaný znak,
  - program č. 7: Bežiaci text "Marian Placko",
- vyslanie reťazca znakov na port RS232 − bude vysvetlená sériová komunikácia rozhrania UART mikrokontroléra s rozhraním RS232 v PC. Obsah jednotlivých príkladov bude nasledujúci:
  - program č. 8: Vyslanie reťazca znakov na RS232,
  - program č. 9: Vyslanie reťazca znakov na RS232-ASCII,
  - program č. 10: Operácie s UART-om,
- snímanie vstupov z tlačidiel − bude ukazovať ako jednoducho je možné s mikrokontrolérom realizovať snímanie vstupov od užívateľa pomocou tlačidiel. Obsah jednotlivých príkladov bude nasledujúci:
  - program č. 11: Operácie pri stlačení tlačidla,

- program č. 12: Po stlačení tlačidla sa rozsvieti príslušná LED dióda,
- program č. 13: Po stlačení tlačidla sa rozsvieti príslušná LED dióda (100 ms). Max. jedna LED dióda svieti,
- program č. 14: Po stlačení tlačidla sa rozsvieti príslušná LED dióda. Sú povolené i kombinácie,
- program č. 15: Po stlačení tlačidla sa rozsvieti príslušná LED dióda (100 ms) a potom zhasne. Sú povolené i kombinácie,
- program č. 16: Po stlačení tlačidla sa rozsvieti príslušný segment na sedem segmentovom displeji (od 0 do 7),
- program č. 17: Počítadlo. Tlačidlo 0 = Štart, tlačidlo 7 = Storno,

➢ maticová klávesnica − bude rozobraná problematika pripojenia maticových klávesníc k mikrokontroléru. Obsah jednotlivých príkladov bude nasledujúci:

- program č. 18: Obsluha maticovej klávesnice 4x3 bez prerušenia, bez možnosti kombinácie tlačidiel. Príslušná LED dióda svieti do stlačenia ďalšieho tlačidla,
- program č. 19: Obsluha maticovej klávesnice 4x3 s použitím prerušenia, bez možnosti kombinácie tlačidiel. Príslušná LED dióda svieti do stlačenia ďalšieho tlačidla,

➢ A/D prevodník (voltmeter) − nastolí problematiku A/D prevodníka, ktorý je obsiahnutý v použitom mikrokontroléry AT90S8535. Obsah jednotlivých príkladov bude nasledujúci:

- program č. 20: Realizácia voltmetra so základným rozsahom 0-5 V, pomocou A/D prevodníka. Zobrazenie na LCD,

➢ zbernica I^2C - AT24C02 (sériová pamäť EEPROM) − bude pojednávaná problematika v spotrebnej elektronike vo všeobecnosti veľmi využívanej sériovej zbernice I^2C. Obsah jednotlivých príkladov bude nasledujúci:

- program č. 21: Zápis dát do EEPROM pamäti 24C02,
- program č. 22: Prečíta 32 bajtov dát z EEPROM pamäti 24C02 a zobrazí na LED,
- program č. 23: Prečíta 32 bajtov dát z EEPROM pamäti 24C02 a zobrazí na LCD,

- zbernica MicroWire - TLC549 (meranie napätia) − bude pojednávať opäť o sériovej zbernici a to o zbernici MicroWire. Ďalej bude zoznamovať s A/D prevodníkom TLC549, ktorý komunikuje práve cez protokol MicroWire. Obsah jednotlivých príkladov bude nasledujúci:
  - program č. 24: Meranie napätia, rozsah 0-5 V, TLC549. Zobrazenie - LED diódy,
- zbernica 1-Wire - DS18B20 (meranie teploty) − bude do tretice pojednávať o protokole pre sériovú zbernicu a to o zbernici 1-Wire. Ďalej bude oboznamovať s teplotným snímačom pracujúcim na tomto protokole. Bude to snímač firmy DALLAS Semiconductor DS18B20. Obsah jednotlivých príkladov bude nasledujúci:
  - program č. 25: Realizácia teplomera s DS18B20 (1-Wire Bus).
- ovládací panel (ovládanie vybraných aplikácii - OS) − bude riešiť centrálne ovládanie vybraných príkladov aplikácií, t.j. mikrokontrolér bude naraz v sebe obsahovať viaceré príklady a výber jednotlivého príkladu bude realizovaný prostredníctvom maticovej klávesnice po zadaní príslušného kódu príkladu. Výpis programu bude mať číslo 26.

Použité prístroje, zariadenia, pomôcky:

- pracovná doska, univerzálne pole − model GB3-354,
- elektronické súčiastky (viď. *Príloha 7*),
- digitálny merací prístroj − RTO DMM-3800-21,
- logická sonda (vlastná výroba),
- napájací adaptér − BYSTRIČAN ZN 9/300 (9 V/300 mA),
- mikrospájka − SOLOMON SL-20,
- osobný počítač − procesor: 1,4 GHz, pamäť RAM: 512 MB,
- drobný materiál − kliešte, svorky, tenký drôt a pod.

Programové (softvérové) vybavenie

V diplomovej práci je pre vypracovanie zadania použité nasledovné programové vybavenie:

- OrCad Release 9 − obsahuje súbor programov pre kreslenie elektronických schém a návrh plošných spojov,
- CodeVisionAVR v.1.23.6a − program slúžiaci na programovanie v jazyku C pre mikrokontroléry ATMEL AVR,
- MS Office 2003 − súbor programov ako je Word − textový editor.

Realizácia diplomovej práce bude vykonaná na katedre KEA MF SPU.

4 VÝSLEDKY PRÁCE

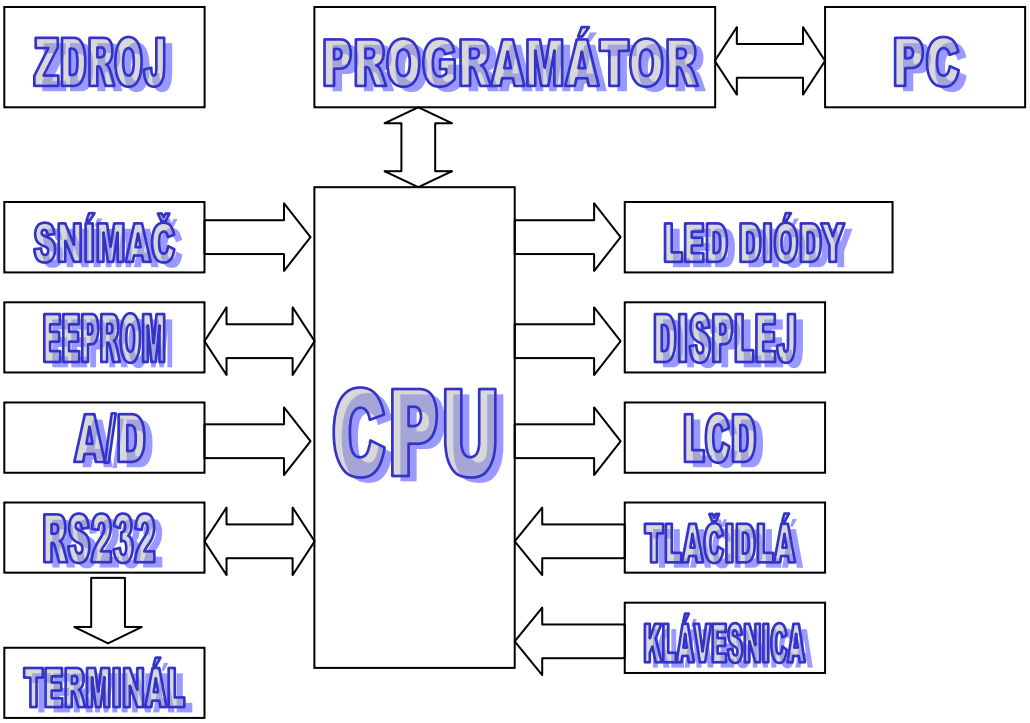
4.1 ÚVOD

Popisované vývojové pracovisko je koncipované ako pomôcka na výučbu jednočipových mikropočítačov veľmi populárneho radu mikrokontrolérov AVR a zároveň ako meracie pracovisko. Je postavené na univerzálnej doske tak, aby sa dalo ľahko doplniť o ďalšie moduly, ktoré sprostredkujú snímanie údajov zo senzorov, silové výstupy na akčné členy a pod. Preto je vybavené ako analógovými, tak číslicovými vstupmi a výstupmi. Jeho softvérové vybavenie je vlastne jednoduchý operačný systém, ktorý reaguje na povely zadávané z klávesnice, podobne ako vybavenie väčšiny priemyslových terminálov.

Vzhľadom k tomu, že naprogramovanie mikrokontroléra je veľmi rýchle a vďaka implementácii ISP metódy programovania i veľmi pohodlné, je možné s ním vyvíjať i akékoľvek ďalšie aplikácie, od jednoduchého "blikača" až po digitálne spracovanie zvuku. Preto sa dobre hodí do výučbového procesu, kde s ním môžu študenti experimentovať.

Architektúra AVR je veľmi univerzálna, čo dokazujú integrované časovače/čítače, záchytné registre, sériový kanál, prioritný systém prerušenia, analógový komparátor, Watchdog, viacvstupový A/D prevodník, čítače pre impulzne šírkovú moduláciu (PWM) atď.

4.2 BLOKOVÁ SCHÉMA ZAPOJENIA

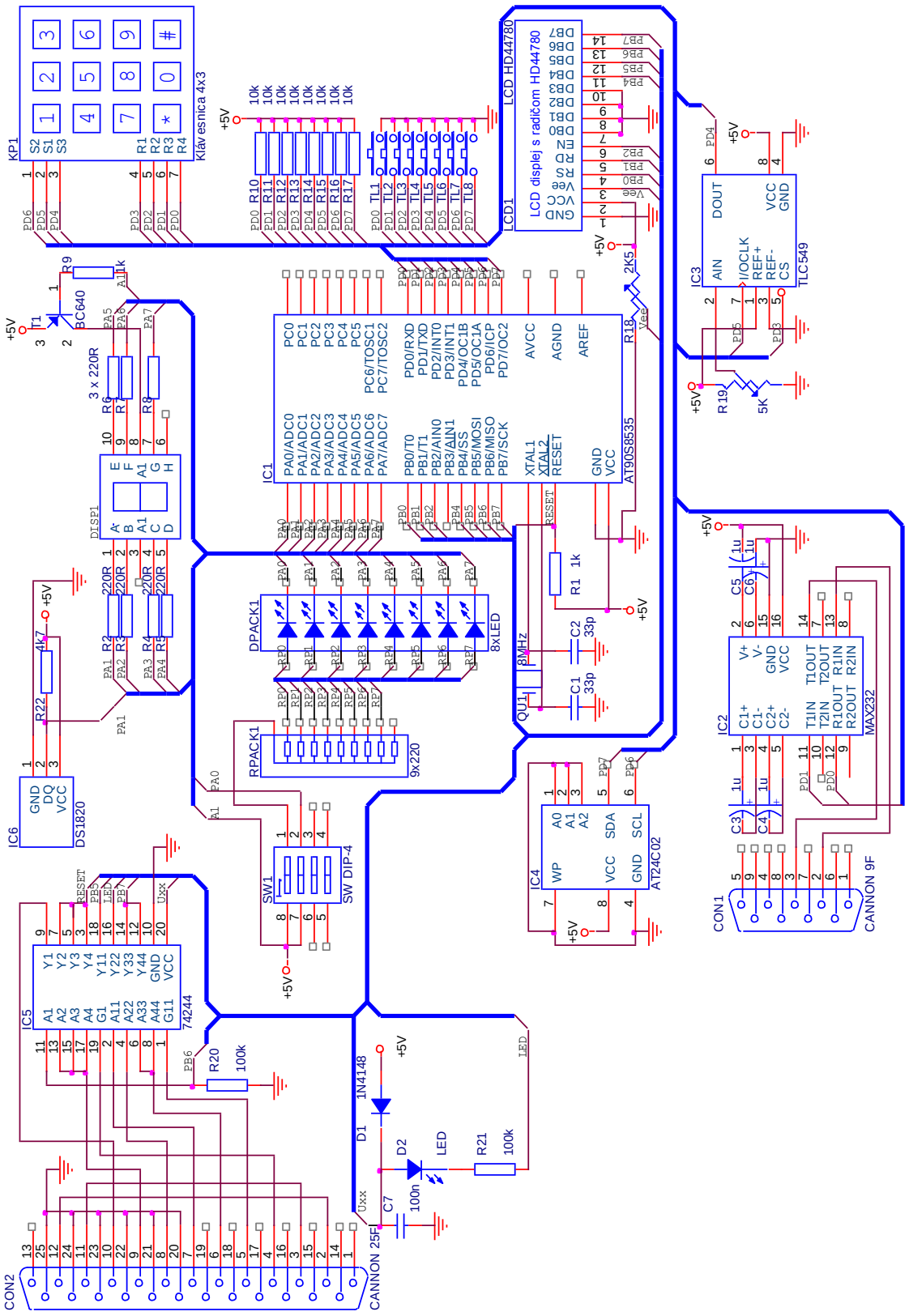


Obr. 17 Bloková schéma zapojenia

Legenda:

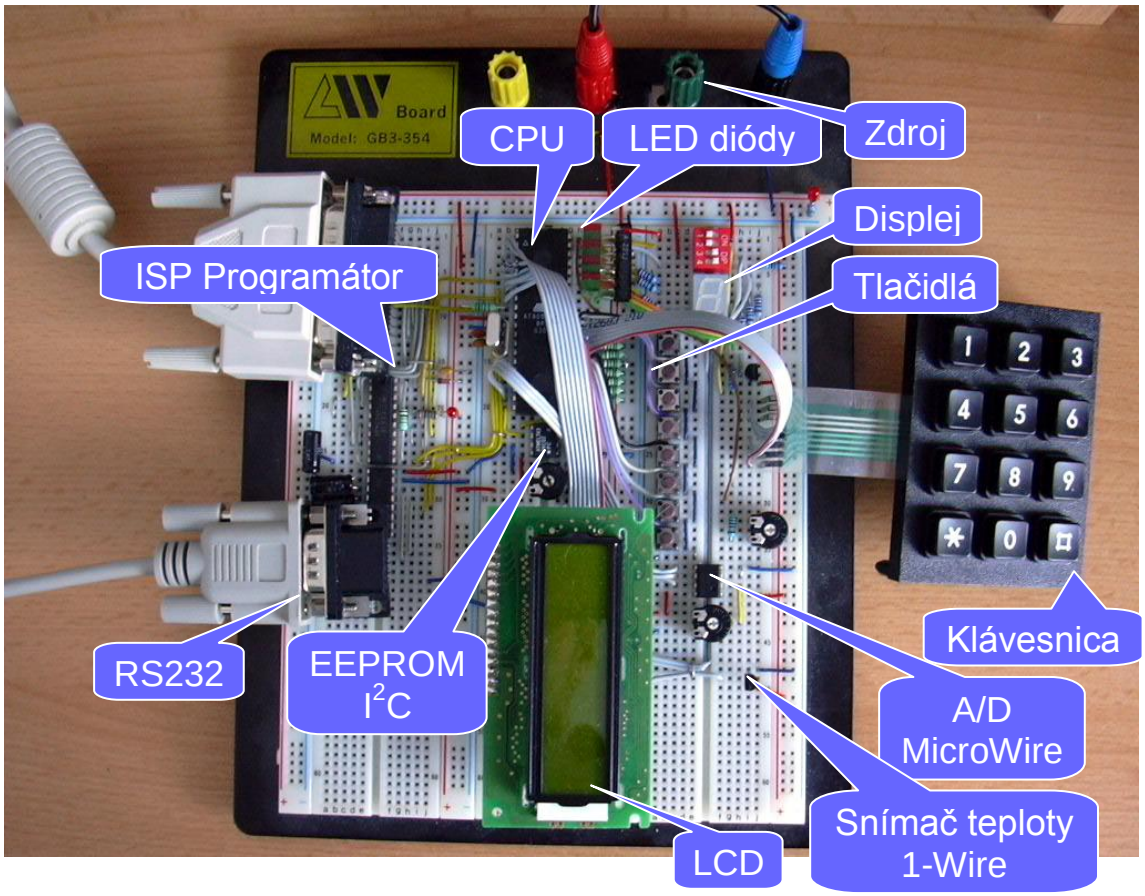
- ZDROJ − napájanie vývojovej dosky 5 V
- PROGRAMÁTOR − zabezpečuje programovanie, verifikáciu a pod. medzi stranou CPU a PC
- PC (Personal Computer) − osobný počítač, ktorý zabezpečuje programové vybavenie pre CPU
- CPU (Central Procesor Unit) − centrálna procesorová jednotka, jadro vývojovej dosky
- LED DIÓDY − pole s LED diódami, ktoré zabezpečuje výstupnú vizualizáciu
- DISPLEJ − 7 segmentový displej, zabezpečuje výstupnú vizualizáciu
- LCD − zobrazovacia jednotka
- TLAČIDLÁ − pole s tlačidlami, ktoré slúžia ako vstup od užívateľa
- KLÁVESNICA − maticová klávesnica na zadávanie vstupu od užívateľa
- SNÍMAČ − pripojenie snímacieho senzora, napr. teplotný senzor DS18S20 a pod.
- EEPROM − externá pamäť napr. AT24C02
- A/D (Analog/Digital) − analógovo číslicový prevodník, napr. pre meranie napätia s TLC 549
- RS232 − sériové rozhranie na komunikáciu, napr. s terminálom na PC a pod.
- TERMINÁL − klientske prostredie na strane PC, ktoré slúži na vizualizáciu dát na PC

4.3 SCHÉMA ZAPOJENIA



Obr. 18 Schéma zapojenia

4.4 VÝVOJOVÁ DOSKA



Obr. 19 Vývojová doska

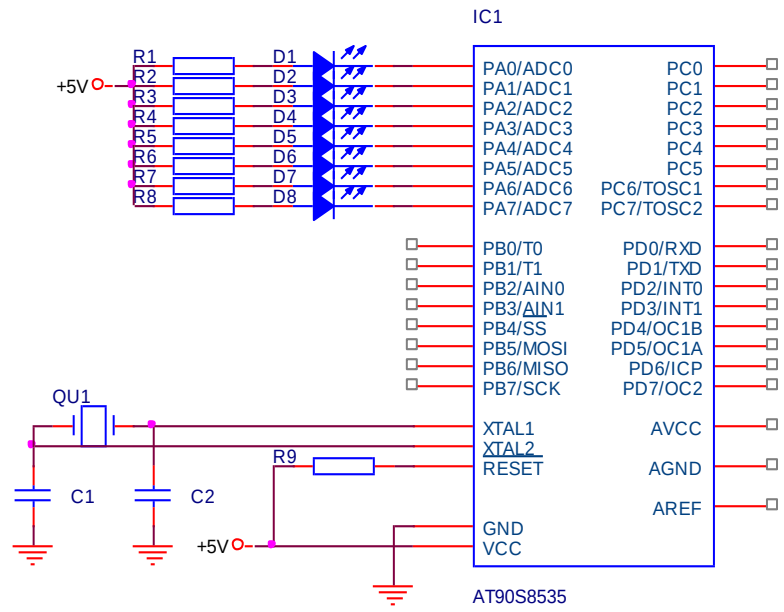
4.5 APLIKÁCIE

- ovládanie LED diód,
- komunikácia s LCD,
- vyslanie reťazca znakov na port RS232,
- snímanie vstupov z tlačidiel,
- maticová klávesnica,
- A/D prevodník – voltmeter,
- zbernica I^2C – AT24C02 (sériová pamäť EEPROM),
- zbernica MicroWire – TLC549 (meranie napätia),
- zbernica 1-Wire – DS18B20 (meranie teploty),
- ovládací panel (ovládanie vybraných aplikácii − OS).

4.5.1 OVLÁDANIE LED DIÓD

Učebnice programovacích jazykov ako prvú ukážku programu v príslušnom jazyku najčastejšie začínajú programom "Ahoj svet" (Hello World), ktorý nerobí nič iné, ako vypísanie nápisu na obrazovku a to podľa charakteru jazyka a prostredia, pre ktorý je príslušný program určený, najčastejšie v režimu príkazového riadku (konzoly), okna či www stránky. V prípade programovania mikrokontrolérov, ako napr. ATMEL AVR, je možnosť začať ešte jednoduchším programom a program "Ahoj svet" si ponecháme až ako druhý program ukazujúci programovanie výstupov na displej LCD.

V prvom programe si ukážeme iba posielanie dát na I/O bránu, napr. PORTA, ku ktorému bude pripojených 8 LED diód.



Obr. 20 Principiálne zapojenie mikrokontroléra s LED diódami

V tomto zapojení bude LED svietiť, keď PORTA bude na príslušnom pine mať log. 0, a bude zhasnutá, ak bude tento signál na úrovni log. 1. Hodnota jednotlivých bitov v bajte poslanom na výstup PORTA potom určí, ktoré LED diódy budú svietiť a ktoré nie. Použitie mikrokontroléra na rozsvietenie niekoľko LED diód bez akejkoľvek ďalšej činnosti nie je rozhodne typickou aplikáciou využitia MCU, tak si naprogramujeme aspoň jednoduchý "blikač", napr. striedavé rozsvietenie párnych a nepárnych LED po 500 ms.

```
/******************** Výpis zdrojového kódu č.1 **********************
Opis programu: Blikanie 8 LED diód (párne/nepárne), pripojených
k portu A, pomocou 2 premenných. Frekvencia blikania je 2 Hz.
*****************************************************************/

#include <90s8535.h>
#define xtal 8000000
#include <delay.h>
unsigned char OutLED1;
unsigned char OutLED2;

void main (void)
{
  DDRA = 0xFF;      // PortA = výstup (000000)
  OutLED1 = 0x55;   // (B=01010101)
  OutLED2 = 0xAA;   // (B=10101010)
  #asm ("cli");     // Zakáže všetky prerušenia (Delay.h)
  while (1)
  {
    delay_ms (500);
    PORTA = OutLED1;
    delay_ms (500);
    PORTA = OutLED2;
  };
}

/*****************************************************************/
```

Zostáva nám ešte popísať zdrojový kód nášho prvého, veľmi jednoduchého programu. Obsahuje jedinú funkciu, čo je v jazyku C povinná funkcia **main,** slúžiaca ako vstupný bod programu. Na začiatku kódu sú do nej ešte pomocou direktívy **#include** vložené hlavičkové súbory **90s8535.h,** čo je súbor obsahujúci definície mien I/O registrov mikrokontroléra AT90S8535 a **Delay.h,** patriaci ďalšej knižnici. K správnej činnosti tejto knižnici je ešte treba definovať konštantu xtal podľa frekvencie použitého kryštálu (v našom prípade 8 MHz) a deklarovať dve pomocné premenné OutLED1 a OutLED2, do nich na začiatku kódu funkciou main vložíme hodnoty určujúce logické úrovne signálu na výstupu PORTA, ku ktorému sú pripojené diódy LED. Predtým samozrejme prikážeme *DDRA=0xff;* t.j. vložíme jednotky do riadiaceho registra portu A, čím tento port definujeme ako výstupný. Ďalšie dva príkazy vkladajú hodnoty 0x55 a 0xAA do pomocných premenných. Tieto dve hodnoty sú dané naším zadaním, keď chceme, aby na výstupe portu A boli striedavo jednotky na párnych a nepárnych pinoch.

Tieto hodnoty obsiahnuté v pomocných premenných budeme na výstup portu A dostávať pomocou príkazu *PORTA=OutLED1;* resp. *PORTA=OutLED2;*. Aby sme zabezpečili požadované blikanie 0,5 s, vložíme medzi tieto príkazy pre výstup na port A ešte príkazy zabezpečujúce časové oneskorenie 500 ms. Pretože požadujeme neustále

blikanie LED diód, budú oba príkazy pre výstup na PORTA i pre vytvorenie časových oneskorení vložené do nekonečnej slučky. Tá je tvorená cyklom **while,** ktorý vyhodnocuje podmienku svojho ďalšieho chodu ako splnenú, lebo jednotka vo *while (1)* sa v jazyku C vyhodnocuje ako **true.**

Takto napísaný program bude po preložení a naprogramovaní do AT90S8535 vykonávať svoju činnosť tak, ako sme si predsavzali, i keď je v programe jedna chyba, spôsobená snahou o čo najväčšiu jednoduchosť a prehľadnosť. Knižnica Delay je postavená tak, že pre získanie požadovaných oneskorení treba zakázať všetky prerušenia. Takže by sme pred nekonečnou slučkou *while(1)* umiestnili *#asm("cli");*.

Podobných chýb sa môžeme vyhnúť pri návrhu pomocou wizardu.

Poznámka: I takto napísaný program by niektorí programátori napísali trochu inak. Napríklad pomocné premenné by definovali už pri deklarácií, či namiesto premenných by použili konštanty. Obdobne i v nasledujúcich programoch sa isto nájdu miesta, ktoré by bolo možné napísať lepšie, stručnejšie, štýlom "skutočných programátorov C", čo obvykle nevedie k prehľadnému a zrozumiteľnému kódu, ktorému sa budeme snažiť dávať prednosť.

Ďalšie modifikácie programu

```
/******************** Výpis zdrojového kódu č.2 **********************
Opis programu: Postupné zapínanie 8 LED diód (0.==>7.) na porte A,
pomocou 8 premenných. Frekvencia blikania je 2 Hz.
*****************************************************************/

#include <90s8535.h>
#define xtal 8000000
#include <delay.h>
unsigned char OutLED_0;
unsigned char OutLED_1;
unsigned char OutLED_2;
unsigned char OutLED_3;
unsigned char OutLED_4;
unsigned char OutLED_5;
unsigned char OutLED_6;
unsigned char OutLED_7;

void main (void)
{
  DDRA = 0xFF;
  OutLED_0 = 0xFE;
  OutLED_1 = 0xFD;
  OutLED_2 = 0xFB;
  OutLED_3 = 0xF7;
  OutLED_4 = 0xEF;
  OutLED_5 = 0xDF;
  OutLED_6 = 0xBF;
  OutLED_7 = 0x7F;
  #asm ("cli");

  while (1)
  {
    PORTA = 0xFF;
    delay_ms (500);
    PORTA = OutLED_0;
    delay_ms (500);
    PORTA = OutLED_1;
    delay_ms (500);
    PORTA = OutLED_2;
    delay_ms (500);
    PORTA = OutLED_3;
    delay_ms (500);
    PORTA = OutLED_4;
    delay_ms (500);
    PORTA = OutLED_5;
    delay_ms (500);
    PORTA = OutLED_6;
    delay_ms (500);
    PORTA = OutLED_7;
    delay_ms (500);
  };
}


/*****************************************************************/
```

```
/******************** Výpis zdrojového kódu č.3 **********************
Opis programu: Postupné zapínanie 8 LED diód (0.==>7.) na porte A,
pomocou 8 rozmerného poľa. Frekvencia blikania je 2 Hz.
*****************************************************************/
#include <90s8535.h>
#define xtal 8000000
#include <delay.h>
unsigned char aLED [8] = {0xFE, 0xFD, 0xFB, 0xF7, 0xEF, 0xDF, 0xBF,
                          0x7F};
char c;  // Počítadlo

void main (void)
{
  DDRA = 0xFF;
  #asm ("cli");
  while (1)
  {
    PORTA = 0xFF;
    delay_ms (500);
    for (c = 0; c < 8; ++ c)
    {
      PORTA = aLED [c];
      delay_ms (500);
    };
  };
}
/*****************************************************************/


/******************** Výpis zdrojového kódu č.4 **********************
Opis programu: "Night Rider" z 8 LED diód na porte A, pomocou
8 rozmerného poľa. Frekvencia blikania je 20 Hz.
*****************************************************************/
#include <90s8535.h>
#define xtal 8000000
#include <delay.h>
unsigned char aLED [8] = {0xFE, 0xFD, 0xFB, 0xF7, 0xEF, 0xDF, 0xBF,
                          0x7F};
char c;

void main (void)
{
  DDRA = 0xFF;
  #asm ("cli");
  while (1)
  {
    for (c = 0; c < 8; ++ c)
    {
      PORTA = aLED [c];
      delay_ms (50);
    };
    for (c = 6; c > 0; -- c)
    {
      PORTA = aLED [c];
      delay_ms (50);
    };
  };
}
/*****************************************************************/
```

4.5.2 KOMUNIKÁCIA S LCD

Naším druhým programom bude už obľúbený program "Ahoj svet" (Hello World), v ktorom na LCD displej zobrazíme jednoduchý nápis.

```
/******************** Výpis zdrojového kódu č.5 **********************
Opis programu: Vyslanie reťazca "Ahoj" na LCD displej pripojený
na PORTB.
*****************************************************************/

#include <90s8535.h>
#asm
  .equ __lcd_port = 0x18; PORTB
#endasm
#include <lcd.h>

void main (void)
{
  lcd_init (16);
  lcd_gotoxy (0,0);
  lcd_putsf ("Ahoj");
  while (1);
}

/*****************************************************************/
```

Na začiatku zdrojového kódu sme umiestnili direktívu vyžadovanú knižnicou LCD a informujúcu, ku ktorému portu bude pripojený LCD. Až potom sme umiestnili direktívu s hlavičkovým súborom lcd.h. V hlavnej funkcií main potom nasleduje inicializácia LCD displeja funkciou *lcd_init(16)*. Príkaz *lcd_gotoxy (0,0);* nastaví aktuálnu pozíciu na LCD displeji na prvý riadok a prvý stĺpec. Riadky i stĺpce sa však počítajú od nuly. Nasleduje príkaz *lcd__putsf ("Ahoj");* vykonávajúci už výstup textového reťazca na LCD. Nasleduje prázdna, nekonečná slučka, o nej sa predpokladá, že do nej programátor vloží nejaký kód. V našom jednoduchom programe žiadny ďalší kód už nepotrebujeme, jediný cieľ a to vypísanie nápisu na LCD je splnený. I tak je nekonečná slučka na konci programu nutnosťou. Preklad z C sa najskôr prevedie do assembleru a i v ňom potom bude na konci nekonečná slučka. Pokiaľ by na konci takáto nekonečná slučka nebola, pokračovalo by po vypísaní nápisu na LCD vykonávanie ďalších strojových inštrukcií. Mikrokontrolér totiž ako kód inštrukcií bude chápať i obsah ďalších pamäťových buniek v programovej pamäti FLASH, umiestených za napísaným programom. Neošetrenie zakončenia programu nejakým definovaným spôsobom, ako napr. je táto slučka, by mohlo spôsobovať problémy!

Ďalšie modifikácie programu

```
/******************** Výpis zdrojového kódu č.6 **********************
Opis programu: LCD – užívateľom definovaný znak.
*************************************************************************/

#include <90s8535.h>
#asm
  .equ __lcd_port = 0x18; PORTB
#endasm
#include <lcd.h>

typedef unsigned char byte;
// Tabuľka pre užívateľom definovaný znak (šípka ukazujúca do pravého
horného rohu)
flash byte char0 [8] =
{
  0B0000000,
  0B0001111,
  0B0000011,
  0B0000101,
  0B0001001,
  0B0010000,
  0B0100000,
  0B1000000
};

// Funkcia použitá k definícii užívateľských znakov
void define_char (byte flash *pc, byte char_code)
{
  byte i, a;
  a = (char_code << 3) | 0x40;
  for (i = 0; i < 8; i ++) lcd_write_byte (a ++, *pc ++);
}

void main (void)
{
 lcd_init (16);
 define_char (char0, 0);
 lcd_gotoxy (0, 0);
 lcd_putsf ("Uzivatelsky znak:");
 lcd_putchar (0);
 while (1);
}


/*************************************************************************/
```

```
/******************** Výpis zdrojového kódu č.7 **********************
Opis programu: Bežiaci text "Marian Placko".
*****************************************************************/

#include <90s8535.h>
#include <delay.h>
#asm
  .equ __lcd_port = 0x18;  PORTB
#endasm
#include <lcd.h>
unsigned char c;

void main (void)
{
  while (1)
  {
    lcd_init (16);
    for (c = 0; c < 29; c ++)
    {
      if (c == 29) c = 0;     // Reset, návrat na začiatok
      lcd_gotoxy (c, 0);
      lcd_putsf ("o");
      lcd_gotoxy (c - 1, 0);  // Zmaže predchádzajúce písmená
      lcd_putsf ("k");
      lcd_gotoxy (c - 2, 0);
      lcd_putsf ("c");
      lcd_gotoxy (c - 3, 0);
      lcd_putsf ("a");
      lcd_gotoxy (c - 4, 0);
      lcd_putsf ("l");
      lcd_gotoxy (c - 5, 0);
      lcd_putsf ("P");
      lcd_gotoxy (c - 6, 0);
      lcd_putsf (" ");
      lcd_gotoxy (c - 7, 0);
      lcd_putsf ("n");
      lcd_gotoxy (c - 8, 0);
      lcd_putsf ("a");
      lcd_gotoxy (c - 9, 0);
      lcd_putsf ("i");
      lcd_gotoxy (c - 10, 0);
      lcd_putsf ("r");
      lcd_gotoxy (c - 11, 0);
      lcd_putsf ("a");
      lcd_gotoxy (c - 12, 0);
      lcd_putsf ("M");
      lcd_gotoxy (c - 13, 0);
      lcd_putsf (" ");
      lcd_gotoxy (0, 1);
      lcd_putsf (" ");

      delay_ms (250);
    }
  };
}

/*****************************************************************/
```

4.5.3 VYSLANIE REŤAZCA ZNAKOV NA PORT RS232

Ako tretí program si uvedieme ďalšiu variantu programu "Ahoj svet" (Hello World), tento krát vyslanie reťazca znakov UART-om cez rozhranie RS232 trebárs do sériového portu počítača PC a zobrazenie tohto nápisu v nejakom terminálovom programe na PC.

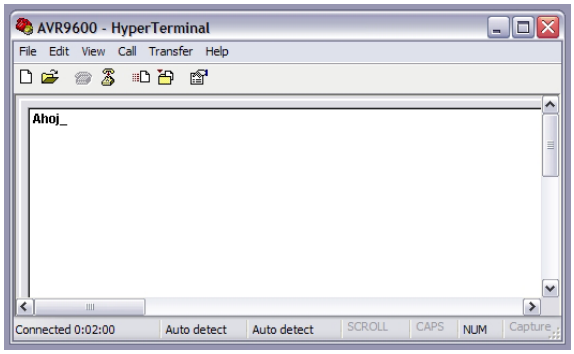
```
/******************** Výpis zdrojového kódu č.8 **********************
Opis programu: Vyslanie reťazca znakov na RS232.
*************************************************************************/

#include <90s8535.h>
#include <stdio.h>

void main (void)
{
  // UART initialization
  // Communication Parameters: 8 Data, 1 Stop, No Parity
  // UART Receiver: Off
  // UART Transmitter: On
  // UART Baud rate: 9600
  UCR = 0x08;
  UBRR = 0x33;

  printf ("Testing...");
  while (1)
  {};
}

/*************************************************************************/
```

Využili sme toho, že **u CodeVisionAVR C je výstupná funkcia printf**, ktorá je súčasťou knižnice štandardných I/O funkcií a vykonáva výstup pomocou UART-u. Preto je potrebné vložiť do stávajúceho sa kódu direktívu *#include<stdio.h>*.

Program preložíme a potom naprogramujeme AT90S8535. K overeniu funkčnosti programu spojíme AVR mikrokontrolér s PC. Predpokladom je, že medzi vývody UART-u, t.j. piny PORTD a konektorom máme obvod, napr. MAX232, prevádzajúci úrovne TTL a RS232. So sériovým portom PC, napr. COM2, spojíme náš RS232 konektor káblom – tzv. nulovým modemom. Vyznačuje sa tým, že vodiče signálov RxD a TxD na vysielacej a prijímacej strane sú vzájomne prepojené, takže napr. vysielané dáta z TxD dosky s AVR prichádzajú do RxD prijímača, t.j. počítača PC. Teraz ešte ostáva spustiť nejaký terminálový program, aby sme mohli prijímať dáta od AVR-ka. Môžeme napr. použiť **Hyperterminál**, ktorý je súčasťou WINDOWS. Spustíme ho výberom z menu **Štart → Programy → Príslušenstvo → Komunikácia**

→ **Hyperterminál**. Program Hyperterminál nám navrhne možnosť vytvoriť nové pripojenie. Potom už nastaveniami postupujeme intuitívne a ostatné zariadi sám WINDOWS.

Ak máme k PC pripojený náš naprogramovaný AVR, môžeme spustiť program s AVR (RESET-om alebo pripojením napätia) a v okne Hyperterminálu uvidíme prijatý text, v našom prípade text "Ahoj".



Obr. 21 Komunikácia cez Hyperterminál

Teraz sa pokúsime program rozšíriť. Viď nasledujúci výpis.

```
/******************** Výpis zdrojového kódu č.9 **********************
Opis programu: Vyslanie reťazca znakov na RS232 - ASCII.
*************************************************************************/

#include <90s8535.h>
#include <stdio.h>
unsigned char c;

void main (void)
{
  // Inicializácia UART-u
  UCR = 0x08;
  UBRR = 0x33;

  printf ("Testing...");
  printf ("\n");
  c = '0';

  while (1)
  {
    putchar (c);
    c ++;
    if (c > 127)
    {
      c = '0';
      printf ("\n");
    }
  };
}

/*************************************************************************/
```

Činnosť programu je zrejmá. Odošle reťazec “Ahoj”, odriadkuje a potom bude v nekonečnej slučke vysielať znaky počínajúc znakom “0” v poradí prislúchajúcej tabuľke ASCII.

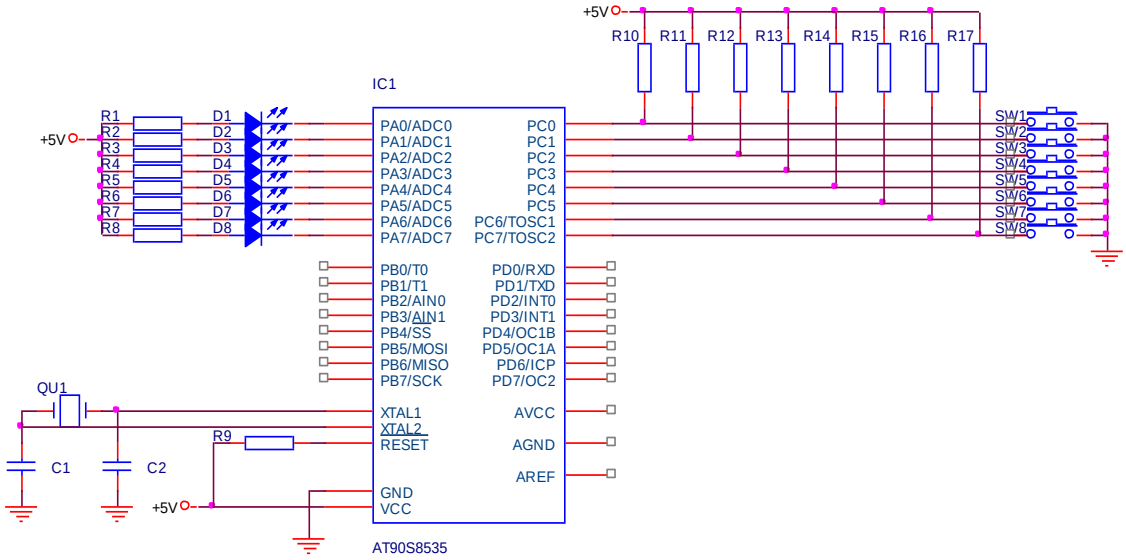
Program je len školský príklad, môže nás inšpirovať k tvorbe užitočnejších aplikácií. Napríklad je možné zvýšiť prenosovú rýchlosť a v nekonečnej slučke namiesto tabuľky ASCII znakov odosielať údaje z A/D prevodníka tohto AVR. Tieto údaje na PC môže namiesto Hyperterminálu zachycovať a spracovávať užívateľský program, ktorý si napíšeme, napr. v Delphi či Visual Basic-u. Možností je veľa.

Ďalšie modifikácie programu

```
/******************** Výpis zdrojového kódu č.10 *********************
Opis programu: Operácie s UART-om.
*************************************************************************/

#include <90s8535.h>
#include <stdio.h>
#define xtal 8000000L
#define baud 9600

void main (void)
{
  char k;

  /* Inicializácia prenosovej rýchlosti UART-u */
  UBRR = xtal/16/baud-1;
  /* Inicializácia riadiaceho registra UART-u RX & TX povolená, bez
      prerušenia, 8 bitov dát */
  UCR = 0x18;

  while (1)
  {
    /* Prijme znak */
    k = getchar();
    /* Vráti znak naspäť */
    putchar(k);
  };
}

/*************************************************************************/
```

4.5.4 SNÍMANIE VSTUPOV Z TLAČIDIEL

V programe č. 1, 2 a 3 sme zatiaľ programovali výstup na PORT, LCD displej či sme výstupné dáta vysielali sériovým UART-om. V ďalších programoch ukážeme, ako naopak vykonávať vstup. Pre školské účely sa obvykle používajú rôzne Start Kit-y, ako napr. STK500, u ktorých vstupom budú tlačidlá, výstupom diódy LED. V ďalšom programe si ukážeme, ako sa dá v jazyku C programovo obslúžiť takýto jednoduchý Start Kit. Predpokladáme, že k PORTC máme pripojené tlačidlá tak, že v pokojovom stave sú na PORTC samé jednotky, po stlačení niektorého tlačidla sa potom na príslušnom pine tohto portu objaví nula. Ako výstup slúžia LED diódy pripojené k PORTA tak, že ak je na niektorom z pinov tohto portu nula, tak LED svieti, ak je tam jednotka, tak nesvieti. Principiálne zapojenie ukazuje *obr. 22.*



Obr. 22 Principiálne zapojenie tlačidiel s LED diódami

Napíšeme jednoduchý program, inšpirovaný ukážkovým programom v asssembleri u Start Kit-u STK500. Jeho úlohou je v nekonečnej slučke zisťovať stav tlačidiel a na stlačenie niektorého z nich nejako reagovať, napr. inkrementáciou (pripočítaním), dekrementáciou (odčítaním), rotáciou a pod. a údaje zobrazovať diódami LED. Viď nasledujúci výpis kódu.

```
/******************** Výpis zdrojového kódu č.11 *********************
Opis programu: Operácie pri stlačení tlačidla.
*************************************************************************/

#include <90s8535.h>
#include <delay.h>
unsigned char c;

void main (void)
{
  PORTC = 0x00;
  DDRC = 0x00;
  PORTA = 0x00;
  DDRA = 0xFF;

  c = 0;
  while (1)
  {
    if (PINC.0 == 0) c += 1;    // Pripočítanie 1
    if (PINC.1 == 0) c -= 1;    // Odčítanie 1
    if (PINC.2 == 0) c << 1;    // Bitový posun o 1 bit doľava
    if (PINC.3 == 0) c >> 1;    // Bitový posun o 1 bit doprava
    if (PINC.4 == 0) c =~ c;    // Bitový doplnok, jednotkový doplnok
    if (PINC.5 == 0) c = 255;   // Nastavenie samých jednotiek
    if (PINC.6 == 0) c = 0;     // Nastavenie samých núl
    if (PINC.7 == 0) c += 1;    // Pripočítanie 1

    PORTA =~ c;  // Inver. 0...LED svieti, 1...LED nesvieti
    delay_ms (100);
  };
}

/*************************************************************************/
```

Činnosť tohto programu je veľmi primitívna. Najskôr sa vykoná inicializácia PORTC ako vstupný a PORTA ako výstupný a nastaví sa hodnota pomocnej premennej na nulu. Potom sa v nekonečnej slučke, začínajúcej príkazom *while(1),* pomocou príkazu *PORTA=~c;* vykonáva zobrazovanie obsahu tejto pomocnej premennej. Chceme, aby sa jednotky v jednotlivých bitoch premennej c prejavili svitom LED, nuly naopak zhasnutím LED. Preto s ohľadom na zapojenie, musíme na výstup PORTA poslať nie obsah premennej c, ale hodnotu vzniknutú z obsahu premennej c negáciou jednotlivých bitov. Preto sme použili operátor ~.

Pred zobrazením hodnoty premennej c ešte môže dôjsť k zmene tejto hodnoty, pokiaľ bude stlačené niektoré tlačidlo. K tomu sme použili príkazy, ako napr. *if (PINC.3 == 0) c = c>>1;* kde PINC.3 sníma logickú úroveň na pine 3 PORTC a interpretuje ju ako nulu či jednotku. Pretože stisk tlačidla sa prejaví nulou, je vykonaný test na stlačenie porovnaním zosnímanej hodnoty s nulou. Stlačenie tlačidla sa prejaví predovšetkým posunutím obsahu premennej c o jedno miesto doprava. Pokiaľ nebude tlačidlo stlačené, nevykoná sa tento príkaz. Takto sa vykonáva postupne test

na stlačenie tlačidiel 0, 1, ... 7. Nakoniec sme ešte zaradili časové oneskorenie 100 ms ako jednoduchý spôsob ošetrenia zákmitov tlačidiel.

Ďalšie modifikácie programu

```
/******************** Výpis zdrojového kódu č.12 *********************
Opis programu:  Po stlačení tlačidla sa rozsvieti príslušná LED dióda.
*********************************************************************/
#include <90s8535.h>
#define xtal 8000000

void main (void)
{
  DDRA = 0xFF;
  PORTA = 0xFF;
  while (1)
  {
    if (PINC.7 == 0) PORTA = 0xFE;
    if (PINC.6 == 0) PORTA = 0xFD;
    if (PINC.5 == 0) PORTA = 0xFB;
    if (PINC.4 == 0) PORTA = 0xF7;
    if (PINC.3 == 0) PORTA = 0xEF;
    if (PINC.2 == 0) PORTA = 0xDF;
    if (PINC.1 == 0) PORTA = 0xBF;
    if (PINC.0 == 0) PORTA = 0x7F;
  };
}
/*********************************************************************/


/******************** Výpis zdrojového kódu č.13 *********************
Opis programu: Po stlačení tlačidla sa rozsvieti príslušná LED dióda
(100 ms). Max. jedna LED dióda svieti.
*********************************************************************/
#include <90s8535.h>
#define xtal 8000000
#include <delay.h>
unsigned char c;

void main (void)
{
  DDRA = 0xFF; PORTA = 0xFF; c = 0;
  while (1)
  {
    if (PINC.7 == 0) c = 0xFE;
    if (PINC.6 == 0) c = 0xFD;
    if (PINC.5 == 0) c = 0xFB;
    if (PINC.4 == 0) c = 0xF7;
    if (PINC.3 == 0) c = 0xEF;
    if (PINC.2 == 0) c = 0xDF;
    if (PINC.1 == 0) c = 0xBF;
    if (PINC.0 == 0) c = 0x7F;

    if (c != 0) PORTA = c;
    delay_ms (100);
    c = 0; PORTA = 0xFF;
  };
}
/*********************************************************************/
```

```
/******************* Výpis zdrojového kódu č.14 **********************
Opis programu: Po stlačení tlačidla sa rozsvieti príslušná LED dióda.
Sú povolené i kombinácie.
*****************************************************************/
#include <90s8535.h>
#define xtal 8000000
unsigned char c0, c1, c2, c3, c4, c5, c6, c7;

void main (void)
{
  DDRA = 0xFF; PORTA = 0xFF;
  c0 = 0xFF; c1 = 0xFF; c2 = 0xFF; c3 = 0xFF; c4 = 0xFF; c5 = 0xFF;
  c6 = 0xFF; c7 = 0xFF;

  while (1)
  {
    if (PINC.7 == 0) c7 = 0xFE;
    if (PINC.6 == 0) c6 = 0xFD;
    if (PINC.5 == 0) c5 = 0xFB;
    if (PINC.4 == 0) c4 = 0xF7;
    if (PINC.3 == 0) c3 = 0xEF;
    if (PINC.2 == 0) c2 = 0xDF;
    if (PINC.1 == 0) c1 = 0xBF;
    if (PINC.0 == 0) c0 = 0x7F;
    // & je bitový logický súčin
    PORTA = c7 & c6 & c5 & c4 & c3 & c2 & c1 & c0;
  };
}
/*****************************************************************/


/******************* Výpis zdrojového kódu č.15 **********************
Opis programu: Po stlačení tlačidla sa rozsvieti príslušná LED dióda
(100 ms) a potom zhasne. Sú povolené i kombinácie.
*****************************************************************/
#include <90s8535.h>
#define xtal 8000000
#include <delay.h>
unsigned char c0, c1, c2, c3, c4, c5, c6, c7;

void main (void)
{
  DDRA = 0xFF; PORTA = 0xFF;
  while (1)
  {
    c0 = 0xFF; c1 = 0xFF; c2 = 0xFF; c3 = 0xFF; c4 = 0xFF; c5 = 0xFF;
    c6 = 0xFF; c7 = 0xFF;
    if (PINC.7 == 0) c7 = 0xFE;
    if (PINC.6 == 0) c6 = 0xFD;
    if (PINC.5 == 0) c5 = 0xFB;
    if (PINC.4 == 0) c4 = 0xF7;
    if (PINC.3 == 0) c3 = 0xEF;
    if (PINC.2 == 0) c2 = 0xDF;
    if (PINC.1 == 0) c1 = 0xBF;
    if (PINC.0 == 0) c0 = 0x7F;
    // & je bitový logický súčin
    PORTA = c7 & c6 & c5 & c4 & c3 & c2 & c1 & c0;
    delay_ms (100);
  };
}
/*****************************************************************/
```

```
/******************* Výpis zdrojového kódu č.16 **********************
Opis programu: Po stlačení tlačidla sa rozsvieti príslušný segment
na sedem segmentovom displeji (od 0 do 7).
*********************************************************************/
#include <90s8535.h>
#define xtal 8000000
unsigned char c;

void main (void)
{
  DDRA = 0xFF; PORTA = 0xFF;
  c = 0;
  while (1)
  {
    if (PINC.7 == 0) c = 0x02;  // 0
    if (PINC.6 == 0) c = 0x3E;  // 1
    if (PINC.5 == 0) c = 0x84;  // 2
    if (PINC.4 == 0) c = 0x0C;  // 3
    if (PINC.3 == 0) c = 0x78;  // 4
    if (PINC.2 == 0) c = 0x48;  // 5
    if (PINC.1 == 0) c = 0x40;  // 6
    if (PINC.0 == 0) c = 0x1E;  // 7

    if (c != 0) PORTA = c;
  };
}
/*********************************************************************/


/******************* Výpis zdrojového kódu č.17 **********************
Opis programu: Počítadlo. Tlačidlo 0 = Štart, tlačidlo 7 = Storno.
*********************************************************************/
#include <90s8535.h>
#define xtal 8000000
#include <delay.h>
unsigned char aSeg [10] = {0x02, 0x3E, 0x84, 0x0C, 0x78, 0x48, 0x40,
                           0x1E, 0x00, 0x08};
char c, i;

void main (void)
{
  DDRA = 0xFF; PORTA = 0x02;
  while (1)
  {
    if ((PINC.7 == 0) || (i == 1)){  // Štart
      // || Disjunkcia (alebo)
      for (c = 0; c < 10; ++ c)
      {
        i = 1;
        PORTA = aSeg [c];
        delay_ms (50);
      };
    };
    if (PINC.0 == 0)
    { // Storno
      i = 0;
      PORTA = 0x02;
    };
  };
}
/*********************************************************************/
```

4.5.5 MATICOVÁ KLÁVESNICA

V niektorých aplikáciách potrebujeme ku komunikácií s MCU viac než osem tlačidiel použitých v predchádzajúcom príklade. Pridanie ďalších tlačidiel by síce bolo možné vyriešiť ich pripojením k ďalšiemu portu MCU a každým z tlačidiel tak ovládať úroveň signálu pre jeden bit portu, ale zbytočne sa tým ochudobňujeme o porty umožňujúce komunikáciu MCU s okolím. Preto pri použití viacej tlačidiel je obvyklé ich usporiadanie do matice a pripojenie tejto tlačidlovej matici k jednému portu.

Programové obslúženie takejto klávesnice je však zložitejšie, ako obsluha samotných tlačidiel. Častým prípadom je použitie 12/16 tlačidiel zapojených do matice 4x3 resp. 4x4. Zdrojový kód aplikácie s klávesnicou v matici 4x3 je uvedený v ďalšej časti.

```
/******************* Výpis zdrojového kódu č.18 ***********************
Opis programu: Obsluha maticovej klávesnice 4x3 bez prerušenia, bez
možnosti kombinácie tlačidiel. Príslušná LED dióda svieti do stlačenia
ďalšieho tlačidla.

MCU: AT90S8535
Pripojenie maticovej klávesnice:
  [PORTD]     [TLAČIDLÁ]     R1
  1 PD3 -----1----2----3----~~~~~~~~~o+5V
             |    |    |     R2   |
  2 PD2 -----4----5----6----~~~~~~
             |    |    |     R3   |
  3 PD1 -----7----8----9----~~~~~~
             |    |    |     R4   |
  4 PD0 -----*----0----#----~~~~~~
          D1 |    |    |
  5 PD5 -|<|-     |    |
          D2      |    |
  6 PD6 -|<|------     |
          D3           |
  7 PD4 -|<|-----------
                          R1..R4=10k..47k
                          D1..D3=1N4148


*****************************************************************/
#include <90s8535.h>
#include <delay.h>
#define KEYIN PINC
void initialization ()
{
  DDRC = 0xFF;   // Vstup
  DDRA = 0xFF;   // Výstup
  PORTA = 0xFF;
  #asm ("cli");  // Zakázanie prerušenia pre "Delay.h"
}
```

```
void keycode ()
{  // Vyhodnotenie stlačenia tlačidla/tlačidiel
  switch (KEYIN)
  {
    case 0xD7 :  // Tlačidlo "1"
      PORTA = 0b11111110;  // Led "1"
    break;
    case 0xB7 :  // Tlačidlo "2"
      PORTA = 0b11111101;  // Led "2"
    break;
    case 0xE7 :  // Tlačidlo "3"
      PORTA = 0b11111011;  // Led "3"
    break;
    case 0xDB :  // Tlačidlo "4"
      PORTA = 0b11110111;  // Led "4"
    break;
    case 0xBB :  // Tlačidlo "5"
      PORTA = 0b11101111;  // Led "5"
    break;
    case 0xEB :  // Tlačidlo "6"
      PORTA = 0b11011111;  // Led "6"
    break;
    case 0xDD :  // Tlačidlo "7"
      PORTA = 0b10111111;  // Led "7"
    break;
    case 0xBD :  // Tlačidlo "8"
      PORTA = 0b01111111;  // Led "8"
    break;
    case 0xED :  // Tlačidlo "9"
      PORTA = 0b01111110;  // Led "1, 8"
    break;
    case 0xDE :  // Tlačidlo "*"
      PORTA = 0b00111100;  // Led "1, 2, 7, 8"
    break;
    case 0xBE :  // Tlačidlo "0"
      PORTA = 0b00011000;  // Led "1, 2, 3, 6, 7, 8"
    break;
    case 0xEE :  // Tlačidlo "#"
      PORTA = 0b00000000;  // Všetky Led
  }
}
void main ()
{
  initialization ();
  while (1)
  {
    PORTC = 0b11011111;  // Stĺpec: 1, tlačidlá: 1, 4, 7, *
    delay_ms (10);
    KEYIN = PORTC;
    keycode ();
    PORTC = 0b10111111;  // Stĺpec: 2, tlačidlá: 2, 5, 8, 0
    delay_ms (10);
    KEYIN = PORTC;
    keycode ();
    PORTC = 0b11101111;  // Stĺpec: 2, tlačidlá: 2, 5, 8, 0
    delay_ms (10);
    KEYIN = PORTC;
    keycode ();
  }
}
/*****************************************************************/
```

Ďalšie modifikácie programu

```
/****************** Výpis zdrojového kódu č.19 *********************
Opis programu: Obsluha maticovej klávesnice 4x3 s použitím prerušenia,
bez možnosti kombinácie tlačidiel. Príslušná LED dióda svieti
do stlačenia ďalšieho tlačidla.
*****************************************************************/
#include <90s8535.h>
#define KEYIN PINC
void initialization ()
{
  DDRC = 0xFF;  // Vstup
  DDRA = 0xFF;  // Výstup
  PORTA = 0xFF;
  TCCR0 = 1;    // Konštanty pre inicializáciu čítača0
  TCNT0 = 0;
  TIMSK = 1;
  #asm ("sei");  // Povolenie prerušenia
}
void keycode ()
{  // Vyhodnotenie stlačenia tlačidla/tlačidiel
  KEYIN = PORTC;
  switch (KEYIN)
  {
    case 0xD7 :  // Tlačidlo "1"
      PORTA = 0b11111110;  // Led "1"
    break;
    case 0xB7 :  // Tlačidlo "2"
      PORTA = 0b11111101;  // Led "2"
    break;
    case 0xE7 :  // Tlačidlo "3"
      PORTA = 0b11111011;  // Led "3"
    break;
    case 0xDB :  // Tlačidlo "4"
      PORTA = 0b11110111;  // Led "4"
    break;
    case 0xBB :  // Tlačidlo "5"
      PORTA = 0b11101111;  // Led "5"
    break;
    case 0xEB :  // Tlačidlo "6"
      PORTA = 0b11011111;  // Led "6"
    break;
    case 0xDD :  // Tlačidlo "7"
      PORTA = 0b10111111;  // Led "7"
    break;
    case 0xBD :  // Tlačidlo "8"
      PORTA = 0b01111111;  // Led "8"
    break;
    case 0xED :  // Tlačidlo "9"
      PORTA = 0b01111110;  // Led "1, 8"
    break;
    case 0xDE :  // Tlačidlo "*"
      PORTA = 0b00111100;  // Led "1, 2, 7, 8"
    break;
    case 0xBE :  // Tlačidlo "0"
      PORTA = 0b00011000;  // Led "1, 2, 3, 6, 7, 8"
    break;
    case 0xEE :  // Tlačidlo "#"
      PORTA = 0b00000000;  // Všetky Led
  }
}
```

```
interrupt [TIM0_OVF] void timer0_int (void)
{
  PORTC = 0b11011111;  // Stĺpec: 1, tlačidlá: 1, 4, 7, *
  keycode ();
  PORTC = 0b10111111;  // Stĺpec: 2, tlačidlá: 2, 5, 8, 0
  keycode ();
  PORTC = 0b11101111;  // Stĺpec: 2, tlačidlá: 2, 5, 8, 0
  keycode ();
}
void main ()
{
  initialization (); while (1) {}
}
/*****************************************************************/
```

4.5.6 A/D PREVODNÍK – VOLTMETER

V predchádzajúcich programoch sme programovali obsluhu externých periférií ako sú tlačidlá, klávesnica či LCD displej alebo diódy LED. Jedinou vnútornou perifériou, ktorú sme zatiaľ programovo obsluhovali, bol UART v *programe č. 3*. Teraz si ukážeme ako programovo obslúžiť A/D prevodník, ktorý je vnútornou perifériou mikrokontroléra AT90S8535. Jeho činnosť je ovládaná zápisom do jeho registra **ADCSR**. Činnosť tohto prevodníka je povolená nastavením najvýznamnejšieho bitu (bit 7 tohto registra na 1. Pre tento, siedmy bit je používané označenie **ADEN**. Ďalší, nižší bit (bit 6) je označovaný **ADSC**, jeho nastavením na 1 je naštartovaný prevod A/D. Po dokončení tohto prevodu bude hodnota tohto bitu rovná 0. Pokiaľ prebieha prevod A/D, nemá prípadné zapísanie nuly do tohto bitu nejaký vplyv na prevod. Nastavenie bitu 3, nazývaného **ADIE**, na jednotku znamená povolenie (aktiváciu) prerušenia **ADC Conversion Complete**. K tomuto prerušeniu dôjde po dokončení A/D prevodu. Najnižšie tri bity registra ADCSR určujú deliaci pomer preddeličky, ktorým je delená frekvencia kryštálu mikrokontroléra. Výstupný signál z preddeličky je privedený na hodinový vstup A/D prevodníka. Po dokončení A/D prevodu je získaná nameraná digitálna hodnota uložená v registroch ADCL a ADCH prevodníka A/D. Obsah oboch registrov vytvára obsah ADCW. Ten je už priamo úmerný meranému napätiu. V našom prípade meriame napätie v rozsahu 0 až 5 V. Pretože A/D prevodník v AT90S8535 je 10-bitový a $2^{10} = 1024$, odpovedá toto číslo 5 V, t.j. 5000 mV. Preto musíme nameraný údaj v ADCW vynásobiť konštantou 5000/1024 = 4,88. Tomu odpovedá príkaz:

```
napatie = (int) (ADCW*4.888);
príkazy
      if (napatie < 1000) lcd_gotoxy(1,1);
      if (napatie < 100) lcd_gotoxy(2,1);
      if (napatie < 10) lcd_gotoxy(3,1);
```

potom zaisťujú umiestnenie nameranej hodnoty na pozíciu LCD, nezávisle na počte cifier nameraného napätia – zarovnanie doprava. Zdrojový kód je veľmi jednoduchý. Viď nasledujúci výpis programu.

```
/******************* Výpis zdrojového kódu č.20 ********************
Opis programu: Realizácia voltmetra so základným rozsahom 0-5 V,
pomocou A/D prevodníka. Zobrazenie na LCD.

Pozn. 1.: JS merané napätie 0 až 5 V sa cez odpor cca 1 kΩ pripojí
+ na pin PA0 a - na GND pin.
*****************************************************************/

#include <90s8535.h>
#include <delay.h>
#asm
  .equ __lcd_port = 0x18; PORTB
#endasm
#include <lcd.h>
#include <stdlib.h>
#define ADC_VREF_TYPE 0x00
unsigned char pom [5];
unsigned int napatie;

interrupt [ADC_INT] void adc_isr (void)
{
  lcd_gotoxy (0, 1);
  lcd_putsf("        ");
  napatie = (int)(ADCW * 4.888);  // 0.005 kvôli zaokrúhleniu
  itoa (napatie, pom);
  lcd_gotoxy (0, 1);

  if (napatie < 1000) lcd_gotoxy (1, 1);
  if (napatie < 100) lcd_gotoxy (2, 1);
  if (napatie < 10) lcd_gotoxy (3, 1);
  lcd_puts (pom);

  delay_ms (20);
  ADCSR |= 0x40;
}

void main (void)
{
  lcd_init (16);
  lcd_gotoxy (0, 0);
  lcd_putsf ("Voltmeter");

  // ADC inicializácia
  ADCSR = 0x8E;    // ADC Interrups: On
  #asm ("sei");
  ADMUX = 0;       // Výber vstupu ADC vstup 0
  ADCSR |= 0x40;   // Začiatok prvého prevodu, jeho dokončenie vyvolá
                       prerušenie ADC

  while (1);
}

/*****************************************************************/
```

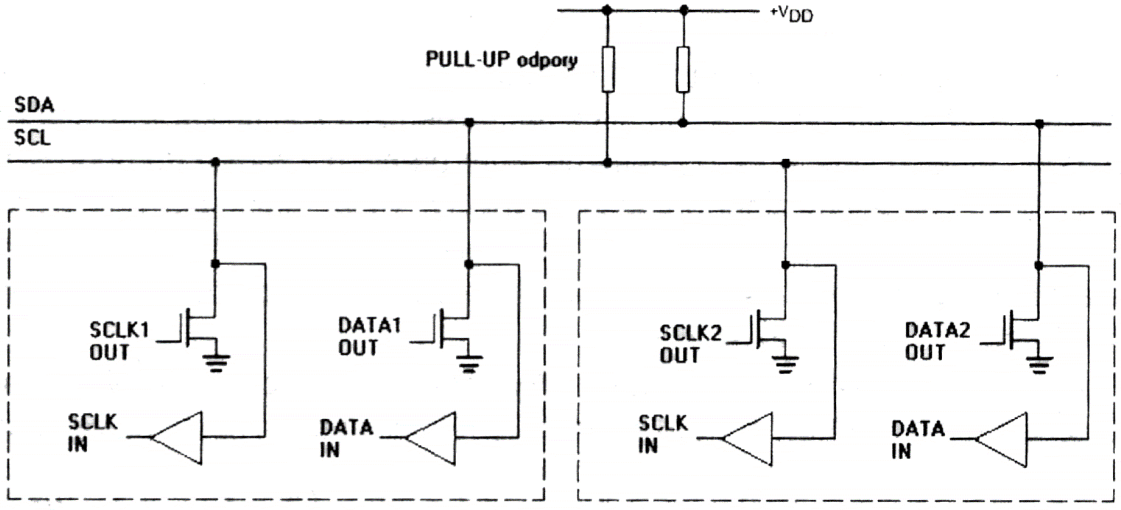
Kód programu je zámerné veľmi jednoduchý, rovnako ako kód ostatných programov v tejto práci. Má sa tým uľahčiť pochopenie jeho funkcie začiatočníkom.

Môže sa však stať základom zložitejších projektov doplnených napr. o komfortnejšiu obsluhu voltmetra, jeho prepojenia s PC pomocou RS232 umožňujúci spracovanie nameraných hodnôt na osobnom počítači. Dokonca je možné takto realizovať i osciloskop.

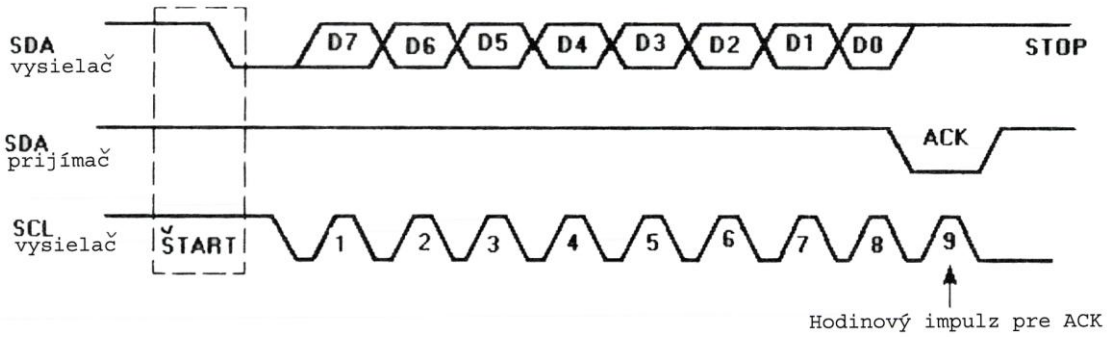
4.5.7 ZBERNICA I²C – AT24C02 (SÉRIOVÁ PAMÄŤ EEPROM)

Zápis

Z hľadiska univerzálnosti použitia, má najväčší praktický význam zbernica I²C (Inter-Integrated-Circuit Bus) vytvorená firmou Philips. Podporu tejto zbernice však nájdeme i u integrovaných obvodov iných výrobcov. Zbernica je tvorená dvojicou vodičov, na ktorých je v pokojovom stave cez pull-up odpory nastavená logická jednotka, viď *obr. 23.*



Obr. 23 Princíp zapojenia zbernice I²C



Obr. 24 Časovanie zbernice I^2C

Jeden z vodičov, označený SCL, prenáša hodinový signál. Druhý, označený SDA, slúži k synchrónnemu prenosu dát *obr. 24.*

Budiče s otvoreným kolektorom umožňujú, aby zbernica bola využívaná skupinou rovnoprávnych radičov zbernice I^2C (multimaster konfigurácia). Odpovedajúci arbitrážny protokol, ktorý rieši pridelenie zbernice pri súčasnej požiadavke viacej radičov zbernice I^2C je popísaný v dokumentácii k tomuto protokolu. V nasledujúcich dvoch programoch budeme uvažovať najčastejší prípad, keď zbernicu bude riadiť len jeden radič zbernice. Bude ním mikrokontrolér AVR riadený programom napísaným v CodeVisionAVR C a využívajúci jeho knižničné funkcie pre I^2C.

Komunikáciu zahajuje radič prvkom ŠTART − zostupná hrana signálu SDA pri jednotke signálu SCL. Ďalej je vyslaný osembitový znak po vodiči SDA počínajúc najvýznamnejším bitom. Vyslaný znak je potvrdený prijímačom stiahnutím signálu SCL na úroveň nula − signál ACK (acknowledge).

Norma I^2C, na rozdiel od MicroWire, definuje i formát prenesených dát, ich potvrdenie a predávanie riadenia medzi účastníkmi komunikácie. Prvým znakom vyslaným radičom je vždy riadiace slovo. Jeho sedem najvýznamnejších bitov A7, A6 až A1 slúži k špecifikácii podriadeného obvodu, s ktorým bude prebiehať komunikácia. Najmenej významný bit A0 riadiaceho slova potom určuje smer nasledujúcej komunikácie, t.j. smer druhého, popr. ďalších slov. A0 = 0 znamená smer prenosu z radiča do podriadeného obvodu, A0 = 1 obrátený smer prenosu. Prvé štyri (popr. ďalšie) bity riadiaceho slova A7, A6, A5, A4 sú určené typom obvodu. Ďalšie bity A3, A2, A1 sú nastavené adresnými vývodmi podriadeného obvodu. Týchto podriadených obvodov teda môže byť k zbernici I^2C pripojených až $2^3 = 8$.

Pre aplikácie v oblasti spotrebnej elektroniky a telekomunikácií vyrábajú firmy Philips a Siemens veľa obvodov riadených zbernicou I^2C. Pre ilustráciu si niektoré uvedieme:

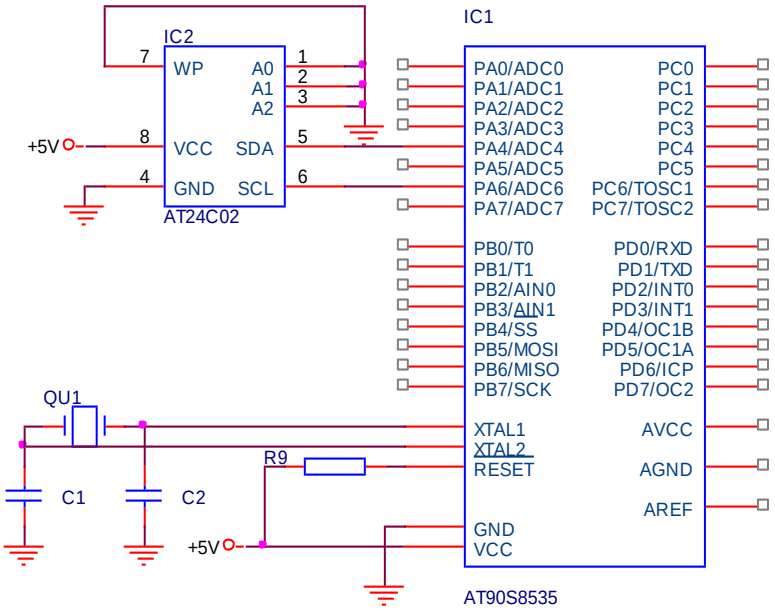
Tab 3. Obvody pracujúce na zbernici I^2C

		A7	A6	A5	A4	A3	A2	A1
PCF8574	8-bitový vstup/výstup	0	1	0	0	a	a	a
PCF8577	64-segmentový radič LCD	0	1	1	1	0	1	0
PCF8578	Radič bodového LCD	0	1	1	1	1	0	a
PCF8582A	Pamäť EEPROM 256x8 b	1	0	1	0	a	a	a
PCF8583	Hodiny/kalendár a pamäť RAM	1	0	1	0	0	0	a
PCF8591	8-bitový A/D, D/A prevodník	1	0	0	1	a	a	a
SAA3028	Infra prijímač kódu RD5	0	1	0	0	1	1	0
TDA8444	Osemnásobný D/A prevodník	0	1	0	0	a	a	a
TSA5055	Syntezátor frekvencie – PLL do 2,6 GHz	1	1	0	0	0	a	a

Pretože obvodov riadených I^2C je veľký počet, nie je možné, aby každý takýto obvod mal v tejto tabuľke svoj riadok a preto niektoré obvody majú rovnaké hodnoty bitov A7, A6, A5, A4. Ide však pritom o obvody s obdobnou funkciou. Napríklad všetky pamäti EEPROM majú hodnoty týchto bitov 1010, takže napr. AT24C01 či 24C02 sú riadené a programované rovnako, ako PCF8582A. Tieto pamäti nám taktiež poslúžia v nasledujúcich dvoch príkladoch k podrobnejšiemu vysvetleniu funkcie protokolu I^2C i ľahkosť naprogramovania mikrokontrolérov AVR ako radiča I^2C pomocou CodeVisionAVR C. Najskôr si ukážeme jednoduchý program umožňujúci naprogramovať niekoľko bajtov v pamäti EEPROM 24C01 či 24C02. Tá bude s MCU AVR prepojená dvomi vodičmi, napr. *obr. 25.*

Jedinou úlohou nasledujúceho programu je naprogramovať do EEPROM 5 znakov abcde t.j. 0x61, 0x62, 0x63, 0x64 a 0x65 do buniek o adresách 0, 1, 2, 3 a 4. Pre splnenie tejto úlohy sme definovali funkciu **zapis** majúcu dva parametre − adresu bunky, do ktorej chceme zapísať jeden bajt dát a hodnotu týchto dát. Telo funkcie **zapis** obsahuje len knižničné funkcie CodeVisionAVR C. Najskôr funkcia *i2c_start()* vygeneruje ŠTART I2C komunikácie. Nasleduje prenos troch bajtov do pamäti EEPROM pomocou funkcie *i2c_write*. Prvý bajt 0xA0 = 10100000 je riadiacim slovom. V ňom prvé štyri bity 1010 informujú, že I^2C komunikuje s pamäťou EEPROM, ďalšie tri bity 000 obsahujú **adresu obvodu,** s ktorým sa komunikuje. V našom prípade sme pripojením pinov A0, A1 a A2 obvodu 24C02 s nulou (GND) zvolili ako túto adresu 000. Dôležité: adresa obvodu daná hardvérovým nastavením pinov A0, A1 a A2 **nemá** vôbec nič spoločného s adresami buniek pamäti EEPROM 0 až 255. Posledným bitom

riadiaceho slova je 0 určujúca, že ďalšie dáta budú posielané do EEPROM. Druhým bajtom preneseným do EEPROM pomocou príkazu *i2c_write(adresa);* je práve adresa bunky EEPROM, do ktorej chceme zapísať dáta a tretím preneseným bajtom pomocou *i2c_write(data);* sú práve dáta, ktoré zapíšeme do EEPROM. Posledný príkaz *i2c_stop();* ukončuje I^2C komunikáciu. Nasleduje už iba príkaz *delay_ms(10);* zabezpečujúci dostatočný čas pre zápis do EEPROM.



Obr. 25 Principiálna schéma spojenia mikrokontroléra so sériovou pamäťou EEPROM prostredníctvom I^2C

```
/******************** Výpis zdrojového kódu č.21 *********************
Opis programu:    Zápis dát do EEPROM pamäti 24C02.
*****************************************************************/
#include <90s8535.h>
#include <delay.h>
#include <ctype.h>
#asm
    .equ __i2c_port=0x12 ;  PORTD
    .equ __scl_bit=6;  SCL 24C01 pin 6
    .equ __sda_bit=7;  SDA 24C01 pin 5
#endasm
#include <i2c.h>

void zapis(unsigned char adresa, unsigned char data)
{
  i2c_start ();
  i2c_write (0xA0);
  i2c_write (adresa);  // Adresa  pamäti EEPROM
  i2c_write (data);
  i2c_stop ();
  delay_ms (10);
}

void main(void)
{
PORTA = 0x00;
DDRA = 0x00;
PORTB = 0x00;
DDRB = 0x00;
PORTC = 0x00;
DDRC = 0xFF;
PORTD = 0x00;
DDRD = 0x40;

TCCR0 = 0x00;
TCNT0 = 0x00;
TCCR1A = 0x00;
TCCR1B = 0x00;
TCNT1H = 0x00;
TCNT1L = 0x00;
OCR1AH = 0x00;
OCR1AL = 0x00;
OCR1BH = 0x00;
OCR1BL = 0x00;
GIMSK = 0x00;
MCUCR = 0x00;
TIMSK = 0x00;
ACSR = 0x80;

  i2c_init ();
  zapis (0x00, 0x61);
  zapis (0x01, 0x62);
  zapis (0x02, 0x63);
  zapis (0x03, 0x64);
  zapis (0x04, 0x65);
  while (1)
  {};
}
/*****************************************************************/
```

Čítanie

Nasledujúci program ukazuje prečítanie prvých 32 bajtov dát z pamäti EEPROM 24C02 pripojenej k AVR MCU rovnako ako v predchádzajúcom príklade. Prečítané dáta sú zobrazené na LCD displeji s 2x16 znakmi, ktorý je pripojený k portu PORTA.

Základom programu je slučka, ktorá sa vykoná 32x. Pri každom priechode touto slučkou, ktorá je riadená parametrom i, sa z pamäti EEPROM z adresy i prečíta obsah na tejto adrese a potom sa zobrazí na LCD. Kód programu je:

```
/****************** Výpis zdrojového kódu č.22 *********************
Opis programu: Prečíta 32 B dát z EEPROM pamäti 24C02 a zobrazí
na LED.
*****************************************************************/
#include <90s8535.h>
#include <delay.h>
#include <ctype.h>
#asm
    .equ __i2c_port=0x12;  PORTD
    .equ __scl_bit=6;  SCL 24C01 pin 6
    .equ __sda_bit=7;  SDA 24C01 pin 5
#endasm
#include <i2c.h>
int i;
unsigned char d;

void main(void)
{
  PORTA=0x00; DDRA=0x00;
  PORTB=0x00; DDRB=0x00;
  PORTC=0x00; DDRC=0xFF;
  PORTD=0x00; DDRD=0x40;
  TCCR0=0x00;  TCNT0=0x00;
  TCCR1A=0x00; TCCR1B=0x00;
  TCNT1H=0x00; TCNT1L=0x00;
  OCR1AH=0x00; OCR1AL=0x00;
  OCR1BH=0x00; OCR1BL=0x00;
  GIMSK=0x00;  MCUCR=0x00;
  TIMSK=0x00;  ACSR=0x80;

  for(i=0;i<32;i++)
  {
    i2c_init();
    i2c_start();
    i2c_write(0xA0);
    i2c_write(i);
    i2c_start();
    i2c_write(0xA1);
    d=i2c_read(0);
    i2c_stop();
    PORTA=d;        // Použijeme LED diódy
    delay_ms(300);  // V prípade použitia LED (zotrvačnosť)
  };
  while (1) {};
}
/*****************************************************************/
```

Pri každom priechode slučkou sa pomocou *i2c_init();* a *i2c_start();* vykoná zahájenie I^2C komunikácie. Potom je do EEPROM pomocou *i2c_write(0xA0)* poslané riadiace slovo 10100000 rovnaké ako v predchádzajúcom príklade. Znamená to, že nasledujúci bajt bude smerovať do EEPROM. Týmto bajtom je adresa i pamäťou EEPROM. Jej prenos zabezpečuje príkaz *i2c_write(i);*. Jeho dôsledkom je, že po potvrdení jeho príjmu signálom ACK sa vykoná nastavenie čítača adries v EEPROM na adresu i. Potom sa vykoná ďalší štart komunikácie I^2C, čítač adries zostane nastavený na adrese i. Potom nasleduje prenos riadiaceho slova 0xA1, čo je 10100001. Pretože posledný bit je 1, je tým určený prenos nasledujúceho bajtu z pamäti EEPROM. Týmto bajtom je práve obsah pamäti EEPROM na adrese i. K čítaniu tohto obsahu slúži príkaz *d=i2c_read(0);* využívajúci funkciu vracajúcu prečítanú hodnotu. Parametrom tejto funkcie je hodnota potvrdenia ACK, čo je 0.

Ďalšie modifikácie programu

```
/******************** Výpis zdrojového kódu č.23 *********************
Opis programu: Prečíta 32 B dát z EEPROM pamäti 24C02 a zobrazí
na LCD.
*****************************************************************/

#include <90s8535.h>
#include <delay.h>
#include <ctype.h>
#asm
  .equ __lcd_port=0x18
#endasm
#asm
    .equ __i2c_port=0x12;  PORTD
    .equ __scl_bit=6;  SCL 24C01 pin 6
    .equ __sda_bit=7;  SDA 24C01 pin 5
#endasm
#include <lcd.h>
#include <i2c.h>

int i;
unsigned char d;

void main (void)
{
  PORTA=0x00; DDRA=0x00;
  PORTB=0x00; DDRB=0x00;
  PORTC=0x00; DDRC=0xFF;
  PORTD=0x00; DDRD=0x40;

  TCCR0=0x00;  TCNT0=0x00;
  TCCR1A=0x00; TCCR1B=0x00;
  TCNT1H=0x00; TCNT1L=0x00;
  OCR1AH=0x00; OCR1AL=0x00;
  OCR1BH=0x00; OCR1BL=0x00;
  GIMSK=0x00;  MCUCR=0x00;
  TIMSK=0x00;  ACSR=0x80;

  lcd_init (16);
  lcd_gotoxy (0, 0);

  for (i = 0; i < 32; i ++)
  {
    i2c_init ();
    i2c_start ();
    i2c_write (0xA0);
    i2c_write (i);
    i2c_start ();
    i2c_write (0xA1);
    d = i2c_read (0);
    i2c_stop ();
    lcd_putchar (d);
  };

  while (1)
 {};
}

/*****************************************************************/
```

4.5.8 ZBERNICA MICROWIRE – TLC549 (MERANIE NAPÄTIA)

V predchádzajúcich príkladoch sme ukázali jednoduché príklady, v ktorých mikrokontrolér AVR komunikuje s okolím prostredníctvom osembitových paralelných portov, alebo pomocou vstavaných periférií, ako je UART alebo A/D prevodník. Ďalšou možnosťou je komunikácia pomocou sériovej zbernice. Ako sériové zbernice obvykle označujeme prepojovacie systémy, ktoré spájajú mikrokontrolér s pomocnými obvodmi v rámci jedného zariadenia. Sériová zbernica šetrí počet vývodov mikrokontroléra a periférnych obvodov, zjednodušuje konštrukciu. Je typicky tvorená dvojicou signálových vodičov. Jeden prenáša hodinový signál, hrany hodinového signálu definujú časové okamžiky, v ktorých sú na druhom vodiči prezentované jednotlivé bity prenášaných dát.

Pre niektoré obvody, ako sú malé pamäti EEPROM (napr. AT24C02) je použitie sériovej zbernice typické. Sériová zbernica je pochopiteľne pomalšia ako zbernica paralelná.

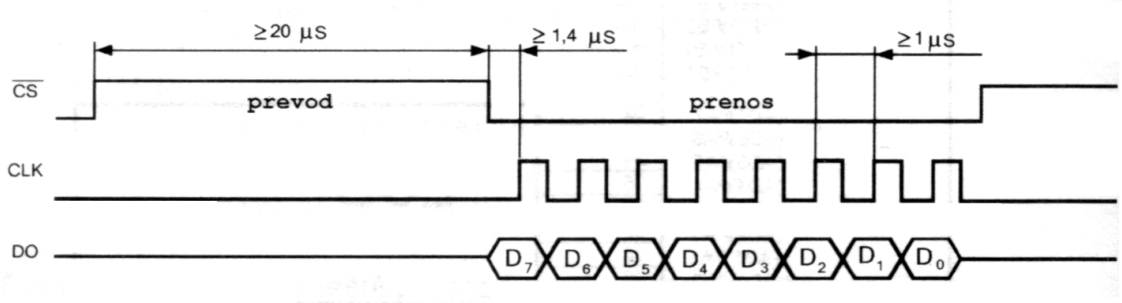
Prítomnosť hodinového signálu určuje, že pôjde o synchrónny prenos. Tento prenos po sériovej zbernici je z obvodového hľadiska jednoduchší ako prenos asynchrónny.

Príkladom jednoduchej sériovej zbernice je zbernica **MicroWire** firmy National Semiconductor. Táto zbernica dovoľuje pripojiť skupinu periférnych obvodov k mikrokontroléru. Je tvorená trojicou vodičov CLK, SO/DI a SI/DO. Hodinový signál CLK riadi prenos po dvoch dátových vodičoch. Prvý prepojuje výstup mikrokontroléra (Serial Out) **SO** so vstupmi (Data In) **DI** periférnych obvodov, druhý prepojuje výstupy (Data Out) **DO** periférnych obvodov na vstup (Serial In) **SI** mikrokontroléra.

K jednému mikrokontroléru je možné pripojiť skupinu periférnych obvodov. Výber periférnych obvodov vyžaduje použitie ďalších výberových vodičov **CS.** Polarita hodinového signálu a časovanie dátových signálov je definované tak, že úrovne na dátových signáloch sa menia vzostupnou hranou hodinového signálu CLK, signály sú čítané s nábežnou hranou hodín CLK. Zbernica MicroWire nemá pevne definovanú dĺžku predávaného slova. Pre väčšinu periférnych obvodov je definovaný určitý primitívny protokol. U jednoduchých periférii však takýto jednoduchý protokol ani nie je potreba a dáta sú predávané ako postupnosť bitov. Takýmto obvodom je 8-bitový A/D prevodník TLC549 od firmy Texas Instruments. Jeho maloobchodná cena je asi 75 SK.

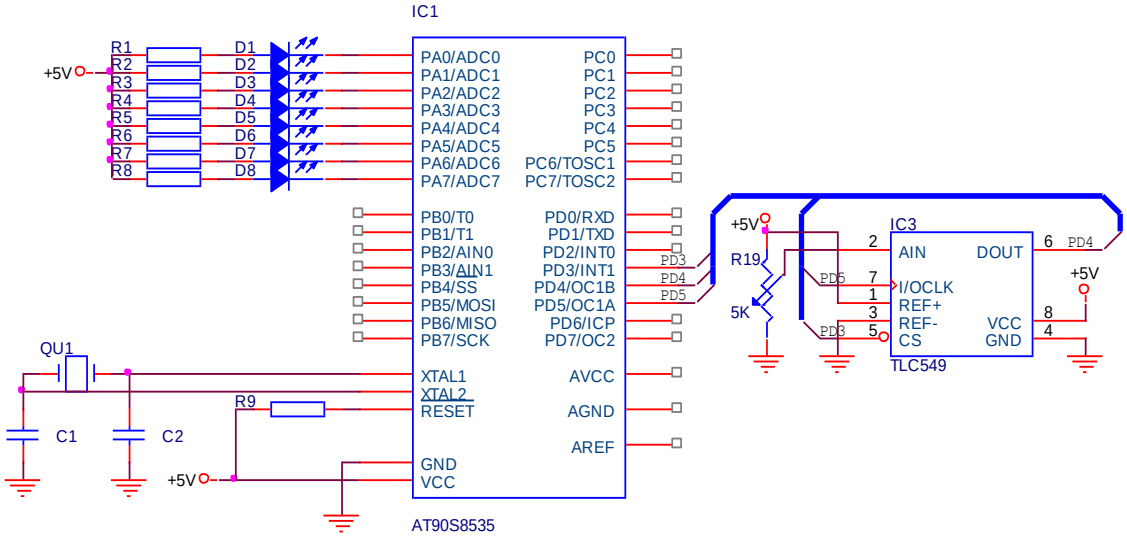
Obvod TLC549 obsahuje posuvný register, do neho umiestňuje nameranú hodnotu. Vlastný prevod trvá menej než 20 ms, takže sú možné i rýchle merania. Signál (Chip Select) CS prepína medzi režimom merania a prevodu. Vlastný prevod sa odohráva pri jednotke na pine CS a trvá 20 ms. Potom je možné na CS priviesť nulu. Týmto sa na dátový vodič privedie bit s najväčšou váhou. Každým hodinovým impulzom sa vysunie bit s nasledujúcou nižšou váhou.

Z popisu funkcie TLC549 je taktiež zrejmé, že dátové signály DI a DO zbernica využíva len ako výstup dát. Priebehy signálov ukazuje nasledujúci *obr. 26*.



Obr. 26 Priebehy signálov zbernice MicroWire u obvodu TCL549

Zdrojový kód v C pre mikrokontrolér AT90S8535 je uvedený nižšie. K zobrazeniu 8-bitových dát použijeme pre jednoduchosť programu len LED diódy. Zjednodušená schéma zapojenia je na *obr. 27*.



Obr. 27 Principiálna schéma A/D prevodníka s obvodom TLC549

```
/******************* Výpis zdrojového kódu č.24 *********************
Opis programu: Meranie napätia, rozsah 0-5 V, TLC549. Zobrazenie LED
diódy.
*********************************************************************/

// TLC549 pripojený na  PORTD
// CS pripojený k PD3, D0 k PD4 , CLK k PD5
// Výstup na PORTA – LED diódy
#include <90s8535.h>
#include <delay.h>
#define cs PORTD.3
#define clk PORTD.5

unsigned char readtlc ()
{
  unsigned char napetie = 0, Ux = 128, loopi = 8;
  clk = 0;
  cs = 0;
  delay_us (2);
  while (loopi --)
  {
    if (PIND.4 == 1)  // Čítanie dát D0
    {
      napetie = napetie + Ux;
    }
    Ux = Ux / 2;
    delay_us (2);
    clk = 1; delay_us (2);   // Hodinový impulz
    clk = 0; delay_us (2);
  }
  cs = 1;
  return napetie;
}

void main (void)
{
  PORTA = 0x00;
  DDRA = 0xFF;
  PORTD = 0x00;
  DDRD = 0x28;  // Pin0 = výstup, pin1 = vstup, pin2 = výstup

  while (1)
  {
    PORTA = readtlc (); // Zobrazenie výsledku na PORTA - LEDky
    delay_ms (100);
  };
}

/*********************************************************************/
```

Program beží v nekonečnej slučke *while (1)*, v ktorej volá funkciu *readtlc ()*, vracajúcu číslo v rozmedzí 0 až 255 v závislosti na nameranom napätí a túto hodnotu posiela na PORTC s pripojenými diódami LED. Činnosť funkcie *readtlc ()* je veľmi jednoduchá. Príkazom *cs = 0;* a nejakou dobou pomocou *delay_us (2);* treba počkať na ustálenie stavu na CS. Hodinový signál bol pritom na 0.

Potom už môžeme pomocou

```
clk = 1;          // Hodinový impulz, nábežná hrana
delay_us (2);
clk = 0;          // Hodinový impulz, dobežná hrana
delay_us (2);
```

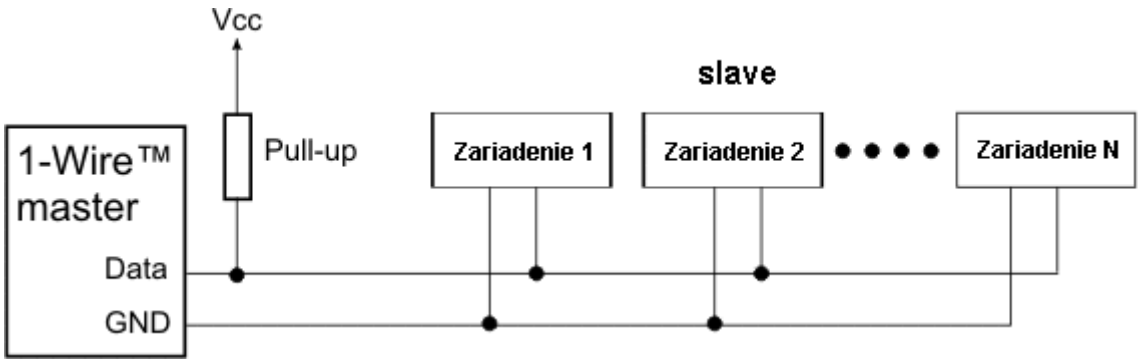
vytvárať hodinové impulzy. Pretože na začiatku je premenná loopi nastavená na osem a impulzy generujeme v slučke *while (loopi --)*, bude počet priechodov touto slučkou a teda počet hodinových impulzov práve osem. Pri každom priechode slučkou bude pred vygenerovaním hodinového impulzu vykonané skontrolovanie úrovne signálu na DO. Ak to bude jednotka, pričíta sa k premennej napätie, odpovedajúce váhe príslušného bitu. Dáta vysiela TLC549 počínajúc najvýznamnejším bitom. V tomto prípade bude táto váha Ux = 128. Pre nasledujúci bit bude táto váha polovičná, pre ďalšie bity štvrtinová, osminová atď. To docielime príkazom *Ux = Ux / 2;* vykonávaným pri každom priechode slučkou.

4.5.9 ZBERNICA 1-WIRE – DS18B20 (MERANIE TEPLOTY)

Zbernica 1-Wire™, navrhnutá firmou Dallas Semiconductor, umožňuje pripojiť niekoľko zariadení k riadiacej jednotke prostredníctvom dvoch vodičov. Táto zbernica je použitá taktiež u technológie iButton, ktorá sa používa v elektronických zabezpečovacích systémoch (dochádzkové systémy a pod.).

Každé 1-Wire™ zariadenie, u ktorého to dáva zmysel (iButton, teplomery, prevodníky apod.), má v sebe pamäť ROM, ktorá obsahuje 64-bitové unikátne číslo, pomocou ktorého je možné jednotlivé zariadenia na zbernici od seba navzájom odlíšiť. Toto číslo sa skladá z typu zariadenia (spodných 8 bitov, kódy sú uvedené v (MAXIM, Application Note 155), sériového čísla (48 bitov) a z CRC kódu (najvyšších 8 bitov). Pomocou tohto čísla je každé zariadenie jednoznačne identifikovateľné. Pri komunikácii s konkrétnym zariadením je potreba po RESET impulze vyslať príkaz Match ROM, potom 64-bitový kód zariadenia, s ktorým sa má pracovať a až potom poslať príkaz.

Na 1-Wire™ zbernici môže byť paralelne pripojených viacej 1-Wire™ zariadení, napr. niekoľko teplotných snímačov DS18B20 spolu s pamäťou DS2431 a A/D prevodníkom DS2450. Výhoda je zrejmá − medzi masterom (napr. PC) a snímačmi vystačíme s dvomi vodičmi.



Obr. 28 Zapojenie zbernice 1-Wire

Algoritmus, akým je možné zistiť identifikačné kódy jednotlivých pripojených zariadení je popísaný v (MAXIM, Application Note 187). Ide o tzv. algoritmus prehľadávania binárneho stromu.

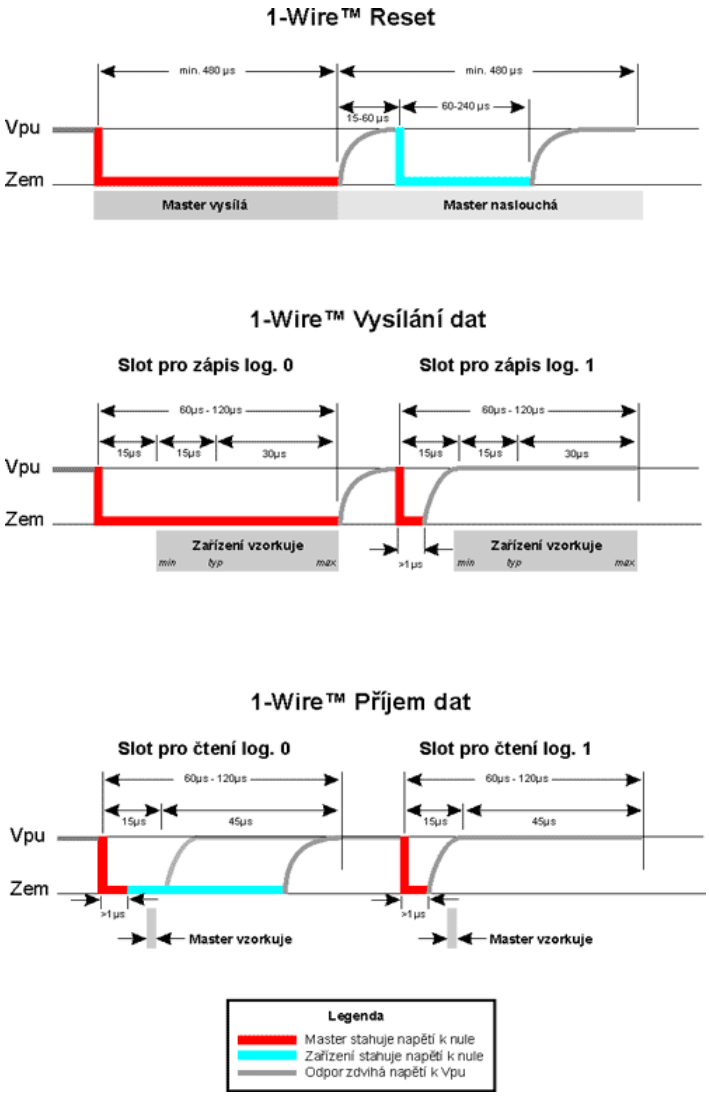
Zbernica má jeden riadiaci obvod (master) a jeden či viacej ovládaných zariadení (slave). Všetky obvody sú zapojené na spoločnú zem a paralelne na spoločný dátový vodič. Tento dátový vodič je pripojený cez odpor cca 5 kΩ na napájacie napätie a "zdvíha" tak zbernicu na úroveň log. 1.

Komunikáciu zahajuje vždy master **reset impulzom**. Najskôr "stiahne" dátový vodič do log. 0 (uzemní ho) a drží ho na tejto úrovni minimálne 480 µs. Potom zbernicu uvoľní a počúva. Odpor zatiaľ vráti zbernicu naspäť na log. 1. Pokiaľ je na zbernici pripojené nejaké 1-Wire zariadenie, tak deteguje tuto vzostupnú hranu a po pauze (15-60 µs), stiahne zbernicu na 60-240 µs k log. 0.

Pokiaľ sa zariadenie správne ohlási, môže master začať vysielať a prijímať dáta. Dáta sú vysielané v tzv. "time slotoch", slovensky by sme skôr povedali v "časových úsekoch" alebo v "oknách". Slot je dlhý 60 až 120 µs a behom jedného slotu je vyslaný alebo prijatý jeden bit informácie. Medzi jednotlivými slotmi musí byť minimálne 1 µs pauza, kde je zbernica v pokojovom stave.

Existujú 4 druhy slotov: Zápis 1, Zápis 0, Čítanie 1 a Čítanie 0. Zápisové sloty slúžia k tomu, aby master vyslal dáta do zariadenia. **Zápis 1** prebieha tak, že master stiahne zbernicu k nule minimálne na 1 µs a maximálne do 15 µs, od začiatku ju opäť uvoľní a ponechá uvoľnenú. Zdvíhací odpor ju teda vytiahne k log. 1. **Zápis 0** je o niečo jednoduchší: Master stiahne zbernicu k 0 a ponechá ju tak po celý slot, teda min. 60 µs. Zariadenie vzorkuje stav na dátovom vodiči zhruba 30 µs po začiatku time slotu.
Čítacie sloty opäť inicializuje master tým, že stiahne zbernicu k nule na minimálne 1 µs a opäť ju uvoľní. Po tomto inicializovaní môže zariadenie vyslať 1 bit buď tým, že

ponechá zbernicu v pokojovom stave (log. 1) alebo ju stiahne (log. 0). Podrobnosti snáď osvetlí nasledujúci obrázok.



Obr. 29 Komunikácia po zbernici 1-Wire. (1-Wire)

Praktická realizácia

Pokiaľ riadi komunikáciu mikrokontrolér, je algoritmus nasledujúci:

Reset

- stiahnuť zbernicu na log. 0, počkať 480 µs,
- uvoľniť zbernicu, počkať 70 µs,
- prečítať zbernicu. Ak je v stave log. 0, je na nej pripojené nejaké zariadenie. Pokiaľ je v log. 1, žiadne zariadenie pripojené nie je,
- počkať 410 µs.

Zápis log. 1

- stiahnuť zbernicu na log. 0, počkať 6 µs,
- uvoľniť zbernicu, počkať 64 µs.

Zápis log. 0

- stiahnuť zbernicu na log. 0, počkať 60 µs,
- uvoľniť zbernicu, počkať 10 µs.

Čítanie

- stiahnuť zbernicu na 0, počkať 6 µs,
- uvoľniť zbernicu, počkať 9 µs,
- prečítať zbernicu. Jej stav udáva prečítaný bit,
- počkať 55 µs.

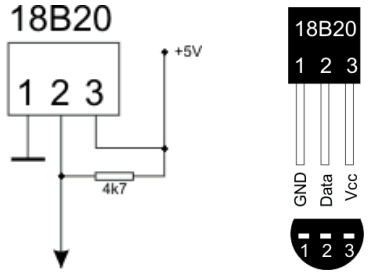
Tieto časy sú doporučené, pokiaľ chceme poznať medzné hodnoty alebo časovanie pre overdrive komunikáciu, doporučujeme (MAXIM, Application Note 126), kde nájdeme i príklady kódu v C.

Komunikácia prebieha po bajtoch a vždy sa vysiela bit 0 (najmenší) ako prvý a bit 7 ako posledný.

Ukážka použitia zbernice 1-Wire − pripojenie teplomera DS18B20

Ako ukážku praktickej aplikácie zbernice 1-Wire sme si pripravili jednoduché zapojenie s obvodom DS18B20, čo je presný digitálny teplomer firmy Dallas. Pripojili sme ho k mikrokontroléru AT90S8535 a pripojili sme iba jeden, takže sme nemuseli prechádzať zbernicu a hľadať ROM kódy.

Zapojenie je veľmi jednoduché a vidíme ho na nasledujúcom obrázku.



Obr. 30 Teplotný snímač DS18B20

```
/******************* Výpis zdrojového kódu č.25 ********************
Opis programu: Realizácia teplomera s teplotným snímačom DS18B20.
Zobrazovanie pomocou LCD.
*******************************************************************/
#asm
    .equ __w1_port = 0x1b
    .equ __w1_bit = 1
#endasm
#asm
    .equ __lcd_port = 0x18
#endasm
#include <lcd.h>
#include <ds1820.h>
#include <delay.h>
#include <math.h>
#include <stdio.h>
```

```
char lcd_buffer [33];
/* Maximálny počet DS1820/DS18S20 pripojených na 1-Wire Bus. */
#define MAX_DEVICES 8
/* Uloženie DS1820/DS18S20 zariadení do pamäte ROM. */
unsigned char rom_code [MAX_DEVICES,9];

main ()
{
  unsigned char i, j, devices;
  int temp;

  lcd_init (16);
  lcd_putsf ("CodeVisionAVR\n1-Wire Bus Demo");
  delay_ms (2000);
  lcd_clear ();

  /* Detekcia počtu DS1820/DS18S20 zariadení pripojených na 1-Wire
      Bus. */
  devices = w1_search (0xf0, rom_code);
  sprintf (lcd_buffer, "%u DS1820\nDevice detected", devices);
  lcd_puts (lcd_buffer);
  delay_ms (2000);

  /* Zobrazenie ROM kódov pre všetky zariadenia. */
  if (devices)
  {
    for (i = 0; i < devices; i ++)
    {
      sprintf (lcd_buffer, "Device #%u ROM\nCode is:", i + 1);
      lcd_clear ();
      lcd_puts (lcd_buffer);
      delay_ms (2000);
      lcd_clear ();
      for (j = 0; j < 8; j ++)
      {
        sprintf (lcd_buffer, "%02X ", rom_code [i, j]);
        lcd_puts (lcd_buffer);
        if (j == 3) lcd_gotoxy (0, 1);
      };
     delay_ms (5000);
    };
  }
  else
  while (1);  /* Stop, ak sa nenašlo žiadne zariadenie. */

  /* Meranie a zobrazenie teploty. */
  while (1)
  {
    for (i = 0; i < devices;)
    {
      temp = ds1820_temperature_10 (&rom_code [i, 0]);
      sprintf (lcd_buffer, "t%u=%i.%u\xdfC", ++ i, temp / 10, abs
      (temp%10));
      lcd_clear ();
      lcd_puts (lcd_buffer);
      delay_ms (800);
    };
  };
}
/*****************************************************************/
```

4.5.10 OVLÁDACÍ PANEL (OVLÁDANIE VYBRANÝCH APLIKÁCIÍ − OS).

Ovládací panel rieši centrálne ovládanie vybraných príkladov aplikácií, t.j. mikrokontrolér v sebe obsahuje viaceré príklady a výber jednotlivého príkladu je realizovaný prostredníctvom maticovej klávesnice po zadaní príslušného kódu príkladu. Viď. *Príloha 1*.

5 DISKUSIA

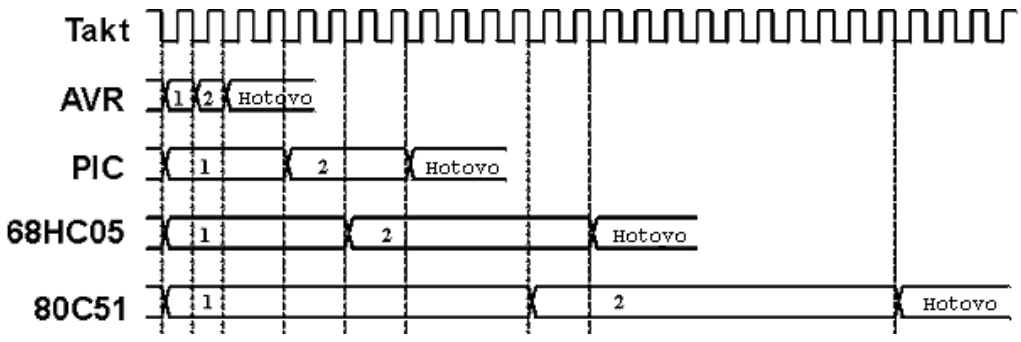
V súčasnej dobe je na trhu veľké množstvo mikrokontrolérov a vývojových prostriedkov od rôznych výrobcov napr.: ATMEL, MICROCHIP, MOTOROLA a ďalších. Vzhľadom na rozmanitý sortiment ponúkaných mikrokontrolérov a vývojových prostriedkov je dôležité si vybrať ten správny pre konkrétnu aplikáciu. Pri zostavovaní diplomovej práce sme vychádzali z týchto zistených skutočností:

1. **Výber architektúry − CISC a RISC architektúra**

Skratky CISC a RISC predstavujú dve rozdielne architektúry CPU (Central Processor Unit) − centrálnej procesorovej jednotky. CISC (Complex Instruction Set Computer) má inštrukčný súbor s takými inštrukciami, ktoré pod jedným operačným kódom vykonajú zložité operácie s variabilitou rôznych adresných módov. Toto sa vykonáva za cenu spracovania týchto inštrukcií v strojových cykloch. Druhá koncepcia RISC (Reduced Instruction Set Computer) je založená na predpoklade, že frekvencia používania niektorých zložitých inštrukcií je tak malá, že sa nevyplatí pre ne plytvať plochou na čipe a v prípade potreby sú nahradené postupnosťou jednoduchých inštrukcií. Inštrukčná sada obsahuje malý počet jednoduchých inštrukcií (cca 30). Vďaka tomu sú jednoduchšie riadiace obvody CPU a skracuje sa doba spracovávania inštrukcií. Riadiace obvody u CISC architektúry zaberajú na čipe približne 60 % miesta, namiesto toho u RISC architektúry je to iba 6-10 % a výkon inštrukcie s výnimkou komunikácie s pamäťou je jeden strojový cyklus (Zmrzlý, 1996).

2. **Výber mikrokontroléra**

- **Výkon (von Neumannová architektúra a Harvardská architektúra)**



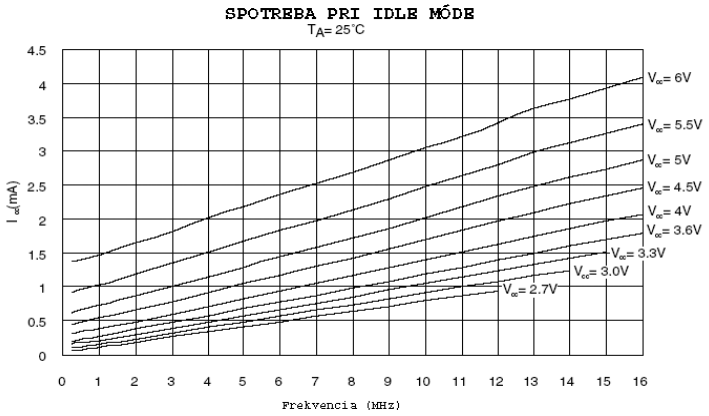
Obr. 31 Porovnanie počtu potrebných taktov na vykonanie dvoch inštrukcií pre známe typy mikrokontrolérov

AVR rodina jednočipových mikrokontrolérov firmy Atmel je vyrobená High-Speed CMOS technológiou s FLASH pamäťou programu, ISP a vyznačuje sa implementáciou RISC inštrukčného súboru. Oproti iným mikrokontrolérom so spoločnou zbernicou pre pamäť a dáta (von Neumannová architektúra) je AVR architektúra charakteristická použitím dvoch oddelených zberníc a pamäťových oblastí pre program a dáta (Harvardská architektúra). Tým je možné nasadenie adresnej a dátovej zbernice s rôznou šírkou. AVR majú 8-bitovú zbernicu pre dáta a 16-bitovú pre inštrukcie (program). Širšou zbernicou pre program je možné prakticky skoro všetky inštrukcie implementovať ako jednoslovné (16-bitový Opcode). Jednostupňová pipeline umožňuje, aby sa počas výkonu inštrukcie následná inštrukcia prenášala z pamäte. V porovnaní s inými mikrokontrolérmi sa rozsah taktovacích frekvencií 0-12 MHz javí pomerne nízky. Ak sa však zohľadní, že frekvencia oscilátora sa vnútorne nedelí a väčšina inštrukcií (až na skokové a Load/Store inštrukcie) sa vykonajú počas jedného taktovacieho impulzu − znamená to výkon až 12 MIPS (Million Instructions per Second).

- **Spotreba (AT90S1200 a PIC16F84)**

Pri získaných poznatkoch sme vychádzali z príslušných datasheet-ov. Podmienky, pokiaľ to bolo možné, boli približne rovnaké (napájanie 5,5 V, externý kryštál 4 MHz a teplota 25 °C), pokiaľ nie je uvedené ináč. Ako referenčných zástupcov sme zvolili mikrokontroléry základného radu a to: AT90S1200 a PIC16F84.

- RUN mód:
  - AVR: cca. 7,2 mA (4 MHz/5,5 V),
  - PIC: typická spotreba 1,8 mA/maximálna 4,5 mA (4 MHz/5,5 V).
- IDLE mód:
  - AVR: 1,5 mA (4 MHz/5,5 V),
  - PIC: nepodporuje.



Obr. 32 Závislosť spotreby od frekvencie a napätia pri IDLE móde (AVR)

- POWER DOWN:
  - AVR: Watchdog On: cca. 30 µA pri 4 V,
    Watchdog Off: < 1 µA pri 4 V (nedá sa určiť z grafu),
  - PIC: Watchdog On: typ 7 µA/max. 28 µA pri 4 V,
    Watchdog Off: typ 1 µA/max. 16 µA pri 4 V.

Zo zistených skutočností sme sa v konečnom dôvode z globálneho pohľadu na príslušný výkon, spotrebu, portfólio firiem, dostupnosť, vývojové prostriedky a budúci rozvoj mikrokontrolérov AVR (AtMega, JTAG a pod.) rozhodli pre mikrokontrolér z rodiny Atmel AVR.

Popisovaná vývojová doska je určená pre RISC-ový procesor z rodiny mikrokontrolérov ATMEL AVR, konkrétne sa jedná o typ AT90S8535 so zabudovaným 10-bitovým A/D prevodníkom. Je navrhnutá tak, aby poskytovala istú mieru univerzálnosti a zároveň nie je na doske napájací zdroj, vlastne sa jedná, akoby išlo o veľkú súčiastku, ktorú aplikujeme podľa vlastnej potreby. Môže poslúžiť ako základ pre Start Kit podobného, ako sú ponúkané na stránkach Atmel-u, ale za podstatne nižšiu cenu.

Mikrokontrolér AVR AT90S8535 skrýva v sebe výkonné jadro, ktoré pri taktovaní kryštálovým oscilátorom 8 MHz dosahuje 8 MIPS, čo je 8000000 inštrukcií za sekundu a to už je čo povedať. V porovnaní, keby sme chceli takúto vysokú rýchlosť dosiahnuť pri známych PIC napr. 16F84, potrebovali by sme ho taktovať kryštálom 32 MHz, pričom výrobca dovoľuje max. 20 MHz. Za spomenutie stojí aj to, že nový rad vyrobených AVR disponuje jadrom, ktoré sa môže taktovať až 16 MHz kryštálom, s ktorým ATMEL dosiahne až 16000000 inštrukcií za sekundu (PIC by potreboval na taký výkon až 64 MHz, čo je nemožné).

Použitý mikrokontrolér AT90S8535 obsahuje:

- FLASH: 8 kB,
- SRAM: 512 B,
- EEPROM: 512 B,
- max. frekvencia: 8 MHz,
- počet inštrukcii: 118,
- počet registrov: 32,
- čítač/časovač: 1x8 bit, 2x16 bit,
- Watchdog,
- A/D prevodník: 8-kanálový, 10-bitový,
- analógový komparátor,
- UART,
- SPI,
- I/O: 32 pinov,
- 1000 programovacích cyklov,
- a ďalšie.

3. Výber metódy programovania

- **ISP (In-System Programming)** – metóda, ktorá je podrobne rozobraná v (AVR In-System Programmer), umožňuje programovanie obvodov priamo v zariadení pomocou vyhradených vodičov určených pre komunikáciu pri programovaní. Obvykle nie je potreba vyššie napájacie napätie ako je menovité napájacie napätie programovaného obvodu. Bežne používané rozhrania pre ISP sú:
  - **SPI (Serial Peripheral Interface)** − rozhranie pre sériovú synchrónnu komunikáciu,
  - **JTAG (Joint Test Actoin Group)** − rozhranie pre diagnostiku a testovanie integrovaných obvodov,
  - **UART (Universal Asynchronous Receiver Transmitter)** − rozhranie pre sériovú asynchrónnu komunikáciu.

  Metódou ISP nie je možné nahrávať (aktualizovať) firmware zariadenia bez toho, aby nedošlo k prerušeniu vykonávania jeho funkcie (programovaný obvod je nutné prepnúť do režimu programovania).

- **IAP (In-Application Programming)** – jedná sa o modifikáciu ISP metódy a umožňuje aktualizáciu firmwaru bez nutnosti prerušiť prácu zariadenia. Pamäť kódu programu je možné zapisovať a čítať priamo z aplikácie (špeciálne inštrukcie mikrokontroléra).

4. Výber programovacieho jazyka

K vývoju aplikácií pre jednočipové mikropočítače je možné použiť celý rad programovacích jazykov (jazyk symbolických inštrukcií − Assembler, C, Pascal, Java, Basic, atď.). Dominantné postavenie si stále udržuje jazyk symbolických inštrukcií s viac než 50 %. Súčasný trend však stále viac smeruje k použitiu vyšších programovacích jazykov, menovite jazyka C.

Pre mnohých vývojárov sa však jedná o zásadný problém, ktorý je možné nazvať strachom z niečoho nového, zložitého a neznámeho. Prečo prechádzať od zavedeného programovania v jazyku symbolických inštrukcií k niečomu inému? Na túto otázku a ďalšie by mal odpovedať nasledujúci text.

Pohľad na vývoj z hľadiska aplikácie

Komplexnosť súčasných jednočipových aplikácií z oblasti automobilového priemyslu, GSM a ďalších oblastí, spolu s narastajúcim tlakom na spracovanie aplikácií jasne smerujú k použitiu vyšších programovacích jazykov.

Výhody použitia vyšších programovacích jazykov pri spracovávaní aplikácií:

- zníženie doby vývoja a tým spojených nákladov,
- sprehľadnenie zdrojových kódov,
- dlhodobá správa a údržba projektu po garantovanú dobu podpory pre dané zariadenie,
- dôraz na multiplatformnosť (jeden zdrojový kód pre rôzne typy mikroprocesorov).

Výrobcovia mikrokontrolérov tento trend podporujú a stále častejšie integrujú väčšie množstvo pamäti RAM a FLASH priamo na čipe. V základnom popise týchto mikrokontrolérov je priamo vyzdvihované použitie v spojitosti s vyššími programovacími jazykmi.

Prečo použiť jazyk C?

Jazyk C bol od prvopočiatku navrhovaný k systémovému (nízko úrovňovému) programovaniu. Vďaka tomu sa presadil všade tam, kde je potreba priamo ovládať ľubovoľný hardvér. V jazyku C je napísaných mnoho operačných systémov, užívateľských a jednočipových aplikácií. Prekladače jazyka C existujú pre väčšinu dostupných typov procesorov.

Výhody jazyka C:

- jednoduchá prenositeľnosť zdrojových kódov programu (multiplatformnosť),
- prehľadnosť zdrojových kódov programu a celkové zjednodušenie pri správe zložitých projektov,
- rýchlejší vývoj aplikácií (použitie štandardných knižníc),
- nástroje na optimalizáciu a validáciu (kontrolu) výsledného kódu programu,
- zavedený programovací jazyk, podporovaný výrobcami hardvéru (mikroprocesorov) a softvéru (kompilátorov).

Nevýhody jazyka C:

- vyššia cena kvalitného vývojového prostredia,
- relatívne väčšie nároky na pamäť dát a programu výslednej aplikácie,
- zložitejšie na osvojenie.

5. Výber periférii (rozhraní):

- výstupná signalizácia (LED, LCD, monitor PC...),
- vstupy od užívateľa (tlačidlá, klávesnica, ...),
- A/D, D/A prevodníky, snímače (teploty, ...), PWM,
- ISP, RS232 a pod.,
- externá pamäť pre ukladanie informácii napr. EEPROM,
- MicroWire,
- 1-Wire,
- I2C,
- USB,
- a ďalšie.

Zaujímavým riešením do budúcnosti bude viac aplikácii postavených na metóde programovania IAP. K aktualizácii firmvéru je možné použiť ľubovoľné rozhrania a tak nahranie aktuálnej verzie programu po lokálnej sieti či z internetu nie je neriešiteľný problém (KIPP, H. Ethernet boot loader).

Z vlastných skúseností môžeme povedať, že pri programovaní mikrokontrolérov rodiny x51 pri použití "klasickej metódy programovania" za použitia programátora (sériového alebo paralelného) popr. v kombinácii s prepínačom (režimy programovania, ladenia a behu aplikácie) bolo nesmierne náročné nielen z pohľadu vývojára, ale opotrebovávali sa napr. nožičky mikrokontrolérov a pod.

V blízkej budúcnosti predpokladáme väčší rozsah využívania mikrokontrolérov ATMEL AVR AtMega, ktoré disponujú perspektívnymi technológiami, ako sú napr.: Bootloader, JTAG a pod.

Bootloader je zavádzač k programovaniu pamäti FLASH a EEPROM. Pamäť FLASH je rozdelená do dvoch častí. Prvú časť tvorí samotná aplikácia a druhou voliteľnou časťou je zavádzač. Veľkosť jednotlivých častí sa nastavuje pomocou konfiguračných bitov BOOTSZ (Boot Size). Čítanie a zápis pamäti vykonáva inštrukcia LPM (Load Program Memory) a SPM (Store Program Memory), ktoré sa vykonajú iba v pamäťovej časti určenej pre zavádzač. Štandardne sa po zapnutí (resetu) obvodu vykonávajú inštrukcie od adresy nula. Zavádzač je však umiestnený na samotný vrchol pamäti FLASH. Je preto nutné nastaviť konfiguračný bit BOOTRST (Boot Reset), čím sa po štarte začne na adrese zavádzača. Ďalšie chovanie je dané iba tým, ako je napísaný kód zavádzača napr. pokiaľ splníme určitú podmienku, dôjde k aktualizácii firmvéru, v opačnom prípade sa spustí aplikácia. Použitie zavádzača však nemôže úplne nahradiť klasický ISP programátor. Zavádzač neumožňuje plnohodnotnú manipuláciu s konfiguračnými a zabezpečovacími bitmi. V plnom rozsahu je možné iba čítať, zápis je obmedzený len na niekoľko vybraných bitov. Vo výsledku to znamená, že konfigurácia mikrokontroléru a nahranie zavádzača musí prebehnúť pomocou paralelného alebo ISP programátora.

Najnovší trend je používanie JTAG rozhrania k ISP programovaniu. JTAG ponúka snáď všetko, čo si môže vývojár embedded (vstavaných) zariadení priať. Bohužiaľ, realita a prax je trochu rozličná. Výrobcom adaptéru JTAG-ICE (In-circuit Emulator) je firma Atmel a cena zariadenia je cca 10500 SK. Na internete sa objavili náznaky, že JTAG-ICE je možné postaviť i v domácich podmienkach (KOSTOMLATSKÝ, M. AVR JTAG ICE − A co na to zlaté (nejen)

české ručičky). Nárast rýchlosti programovania oproti iným riešeniam postavených pre sériový port je zanedbateľný. Problémom naďalej ostáva max. prenosová rýchlosť sériového portu 115200 Bd (z dôvodu kompatibility nie je možné nastaviť vyššiu rýchlosť). Je potreba tiež podotknúť, že JTAG rozhranie sa vyskytuje až u modernejších typov mikrokontrolérov ATMEL AVR.

V neposlednom rade vítame nástup vyšších programovacích jazykov ako napr.: C, Visual Basic, Java a pod., ktoré pri rozsiahlejších projektoch svojou štruktúrou výrazne uľahčia vývojárom ladenie programov pre CPU.

6 NÁVRHY NA VYUŽITIE POZNATKOV

Konštrukcia vychádzala z praktických požiadaviek na výučbu elektrotechnickej praxe, výpočtovej techniky a automatizácie. Pre jej ďalšie využitie sa predpokladá, že si k nej užívateľ postaví patričné prevodníky, silové obvody a snímače podľa vlastných požiadaviek, čo zvlášť na katedre elektrotechnického zamerania, pre ktorú bola vyvíjaná, nemá byť veľký problém. Riešenie úloh v diplomovej práci a jej rozširovanie by malo pomôcť rozvíjať praktické vedomosti študentov a pripraviť ich na styk s praxou.

S použitím pripravených zdrojových programov je možné konštrukciu ľahko adaptovať (preprogramovať) tak, aby bola použiteľná i v inej triede aplikácií, než pre ktorú bola navrhnutá. Snažili sme sa zachovať čo najväčšiu komplexnosť funkcií a ovládania. Preto je vhodné pri jej zaradení do výučby oboznámiť študentov i so spomínaným radom mikrokontrolérov ATMEL AVR podrobnejšie a naučiť ich tieto procesory programovať na úrovni jazyka C. Zdrojové kódy napísané v jazyku C, použité v jednotlivých aplikáciách pre mikrokontrolér ATMEL AVR by niektorí programátori napísali trochu ináč, lepšie, stručnejšie, štýlom "skutočných programátorov C", čo obvykle nevedie k prehľadnému a zrozumiteľnému kódu, ktorému sme dávali prednosť. Preto odporúčame prečítať (MANN, Burkhard. 2003) k tejto konštrukcii. Popisované vývojové pracovisko môže poslúžiť ako základ pre Start Kit podobného, ako sú ponúkané na stránkach ATMEL-u, ale za podstatne nižšiu cenu.

V diplomovej práci je možné vyzdvihnúť:

- jednoduchosť konštrukcie,
- finančnú nenáročnosť,
- rýchly vývoj a experimentovanie,
- pedagogické využitie.

Budúci trend vidíme v masívnejšom presadení sa vyšších programovacích jazykov (C, Visual Basic, Java a pod.). S nástupom nového radu mikrokontrolérov AtMega bude výrazne zefektívnené odlaďovanie aplikácii v podobe rozhraní a technológii:

- JTAG (rozhranie pre diagnostiku a testovanie integrovaných obvodov),
- Bootloader (zavádzač k programovaniu pamäti FLASH a EEPROM),
- IAP (jedná sa o modifikáciu ISP metódy a umožňuje aktualizáciu firmvéru bez nutnosti prerušiť prácu zariadenia).

Pre ďalší vývoj experimentálneho pracoviska je možné sa zamerať i na komerčné produkty Start Kit-ov:

- STK 500/AVR ISP (firma ATMEL),
- Kanda Systems STK200+/300,
- Dontronics DT006,
- Vogel Elektronik VTEC-ISP,
- Futurlec JRAVR,
- MicroTronics ATCPU/Mega 2000,
- a ďalšie.

Zo softvérového riešenia spomenieme produkty:

- CodeVisionAVR (firma HP InfoTech),
- AVR Studio (firma ATMEL),
- WAVRASM (firma ATMEL),
- AVR Edit (firma Tera Bank),
- BASCOM (firma MSC Electronics),
- IAR Embedded Workbench (firma IAR)
- IAR MakeApp for Atmel AVR (firma IAR),
- JAVA/JEPES (firma Mjolner Informatics),
- PASCAL (E-LAB Computers),
- a ďalšie ako Keil a pod.

7 ZÁVER

Diplomová práca bola riešená s cieľom zvýšiť vedomostnú úroveň študentov a podať základný prehľad o problematike mikroprocesorovej techniky. Jej úlohou bolo navrhnúť vývojové pracovisko pre výučbu a experimentovanie.

V dnešnej dobe sa bez znalostí mikroprocesorovej techniky v elektronike a automatizácii už prakticky nedá pracovať. Pracovníkov s týmto zameraním (so strednou i VŠ kvalifikáciou) je stále veľký nedostatok. Slovenské firmy sa opakovane presvedčujú, ako je ťažké získať kvalifikovaného zamestnanca. Podobné problémy majú i v zahraničí. Z tohto dôvodu je dôležité poukázať študentom význam a perspektívu mikroprocesorovej techniky.

Základom nami navrhovaného experimentálneho pracoviska je mikrokontrolér od firmy ATMEL AT90S8535 so zabudovaným 8-kanálovým 10-bitovým A/D prevodníkom a modernými funkčnými blokmi. Mikrokontrolér AT90S8535 je radu AVR, ktorý v sebe skrýva výkonné jadro, ktoré pri taktovaní kryštálovým oscilátorom 8 MHz dosahuje 8 MIPS, čo je 8 000 000 inštrukcií za sekundu. V porovnaní, keby sme chceli takúto vysokú rýchlosť dosiahnuť pri známych PIC napr. 16F84, potrebovali by sme ho taktovať kryštálom 32 MHz, pričom výrobca dovoľuje max 20 MHz.

Na programovanie mikrokontroléra sme zvolili vyšší programovací jazyk C, ktorý na rozdiel od jazyka symbolických adries (assembleru) má pre budúci vývoj v mikroprocesorovej technike veľké prednosti.

Pri výbere jednotlivých aplikácii sme sa inšpirovali podobnými komerčnými Start Kit-mi dodávanými významnými firmami na tomto poli pôsobnosti, ktoré majú dlhoročné skúsenosti v oblasti mikroprocesorovej techniky ako napr.: ATMEL, INTEL, DALLAS MICROCHIP a ďalšie.

V diplomovej práci sa nám podarilo splniť všetky stanovené ciele ako: navrhnúť jednoduché experimentálne pracovisko s možnosťou uplatnenia vo vyučovacom procese, ľahko prispôsobiteľné pre rôzne typy aplikácii, možnosť jednoduchým a hlavne efektívnym spôsobom preprogramovateľné (ISP programovanie) a v neposlednom rade cenovo nenáročné na realizáciu.

V navrhnutom experimentálnom pracovisku boli implementované:

- ISP programovanie,
- zobrazovacie jednotky (LED, 7 segmentový displej, LCD),
- snímanie vstupov (tlačidlá, maticová klávesnica, prerušovací systém),
- rozhranie RS232 na pripojenie k PC,
- rozhranie I^2C pre aplikáciu na komunikáciu s externou pamäťou,
- rozhranie MicroWire pre aplikáciu na meranie napätia,
- rozhranie 1-Wire pre aplikáciu na snímanie teploty.

Pre ďalšie pokračovanie v diplomovej práci by sme navrhovali prejsť na už dnes moderný rad mikrokontrolérov AtMega od firmy ATMEL, ktorý v sebe zahrňuje technológie ako napr.: JTAG, Bootloader a pod. Výrazným medzníkom je i technológia programovania a odlaďovania aplikácii tzv. IAP, ktorá umožňuje aktualizáciu firmvéru bez nutnosti prerušenia práce zariadenia.

Je dôležité si uvedomiť, že jedným z najperspektívnejších oborov súčasnosti i budúcnosti je a bude elektronika a zvlášť tak mikroprocesorová technika. Preto by sa na školách mal dávať dôraz na toto smerovanie. Pomôckou na pochopenie mikroprocesorovej techniky a zvýšenie povedomia o túto problematiku u študentov môže poslúžiť nami navrhnuté experimentálne pracovisko.

8 POUŽITÁ LITERATÚRA

HRBÁČEK, Jiří. 1999. Komunikace mikrokontroléru s okolím − 1. díl. Praha: BEN, 1999, 159 s., ISBN 80-86056-36-8.

HRBÁČEK, Jiří. 2002. Komunikace mikrokontroléru s okolím 2. Praha: BEN, 2002, 152 s., ISBN 80-86056-73-2.

HRBÁČEK, Jiří. 2004. Moderní učebnice programování mikrokontrolérů PIC. Praha: BEN, 2004, 96 s., ISBN 80-7300-136-5.

KAINKA, Burkhard. 1998. Využití rozhraní PC. Měření,řízení a regulace pomocí standardních portů PC. Ostrava-Plesná: HEL, 1998, 133 s., ISBN 80-902059-3-3.

KĽÚČIK, Ján − FRONC, Vojtech. 2001. Mikrokontroléry ATMEL s jádrem 8051. Praha: BEN, 2001, 127 s., ISBN 80-7300-008-3.

MANN, Burkhard. 2003. C pro mikrokontroléry. Praha: BEN, 2003, 280 s.,
ISBN 80-7300-077-6.

MATOUŠEK, David. 2002. Práce s mikrokontroléry ATMEL AT89C2051. Praha: BEN, 2002, 263 s., ISBN 80-7300-094-6.

MATOUŠEK, David. 2002. Práce s mikrokontroléry ATMEL AT89S8252. Praha: BEN, 2002, 304 s., ISBN 80-7300-066-0.

MATOUŠEK, David. 2003. Práce s mikrokontroléry Atmel AVR. Praha: BEN, 2003, 376 s., ISBN 80-7300-088-1.

RATAJ, V. − RYBANSKÁ, M. − JUREKOVÁ, Z. − BOREKOVÁ, B. 2004. Metodika písania záverečných prác na SPU v Nitre. Nitra: SPU v Nitre, 2004, 82 s.,
ISBN 80-8069-328-5.

SKALICKÝ, Petr. 1998. Mikroprocesory řady 8051. 2. roz. vyd. Praha: BEN, 1998, 159 s., ISBN 80-86056-39-2.

STŘÍBRSTÝ, A. a kol. 1994. Technické prostředky pro řízení. Praha: ČVUT.

ŠUBRT, Vladimír. 2002. Mikrokontroléry Atmel AVR - vývojové prostředí. Praha: BEN, 2002, 95 s., ISBN 80-7300-055-5.

VACEK, Václav. 2001. Učebnice programování ATMEL s jádrem 8051. Praha: BEN, 2001, 143 s., ISBN 80-7300-043-1.

VÁŇA, Vladimír. 2003. Mikrokontroléry ATMEL AVR – Programovaní v jazyce C. : popis práce ve vývojovém prostředí CodeVisionAVR C. Praha: BEN, 2003, 215 s., ISBN 80-7300-102-0.

VÁŇA, Vladimír. 2003. Mikrokontroléry ATMEL AVR - popis procesoru a instrukční soubor. Praha: BEN, 2003, 336 s., ISBN 80-7300-083-0.
VÁŇA, Vladimír. 2003. Začínáme s mikrokontroléry Motorola HC08 Nitron. Praha: BEN, 2003, 96 s., ISBN 80-7300-124-1.
Zmrzlý, S. 1996, Mikroprocesorová technika. : skripta VUT FEI Brno. Brno: PC-DIR, 1996.

Firemná literatúra ANALOG DEVICES, MAXIM, ATMEL, INTEL, MICROCHIP, MOTOROLA, DALLAS SEMICONDUCTOR, NATIONAL SEMICONDUCTOR, PHILIPS.
Katalóg elektronických súčiastok, S.O.S electronic s.r.o., 2004/2005.
Součástky pro elektroniku, GM electronic s.r.o., 2004.

1-Wire. http://www.hw.cz/rozhrani/1wire/hw_1wire.html
Atmel Corporation homepage. http://www.atmel.com/
AVR In-System Programmer. http://savannah.nongnu.org/projects/uisp
Code VisionAVR. http://www.hpinfotech.ro
HW Server. http://www.hw.cz
KIPP, H. Ethernet boot loader. http://www.ethernut.de/en/eboot/index.html
KOSTOMLATSKÝ, M. AVR JTAG ICE − A co na to zlaté (nejen) české ručičky. http://www.mcu.cz/modules/news/article.php?mode=flat&storyid=353
KYLE, J. P. STK500 Protocol AVR Bootloader. http://www.avr1.org/stk500boot/stk500boot.html
LANCONELLI, C. PonyProg serial device programmer. http://www.lancos.com/prog.html
MAXIM, Application Note 126. http://pdfserv.maxim-ic.com/en/an/app126.pdf
MAXIM, Application Note 155. http://pdfserv.maxim-ic.com/en/an/app155.pdf
MAXIM, Application Note 187. http://pdfserv.maxim-ic.com/en/an/app187.pdf
MCU server. http://www.mcu.cz
Plačko, Marián. http://www.mplacko.tk
Řehák, Jan. LPT ISP Prog. http://www.hw.cz/constrc/lpt_isp_prog/
The small Word of HC08. http://www.hc08.cz/

9 PRÍLOHY

1. Ovládací panel (ovládanie vybraných aplikácii − OS)

CD

2. Prehľad “rodín” mikrokontrolérov
3. Bloková štruktúra AT90S8535
4. Rozloženie vývodov na AT90S8535
5. Inštrukčná sada pre mikrokontroléry ATMEL
6. Kompilátory a vývojové prostredia pre mikrokontroléry ATMEL AVR
7. Prehľad a opis elektronických súčiastok

PRÍLOHA 1 – OVLÁDACÍ PANEL (OVLÁDANIE VYBRANÝCH APLIKÁCII – OS)

```
/****************** Výpis zdrojového kódu č.26 **********************
© Copyright 2005 Gordon
Web: http://www.gordon.euweb.cz, http://www.mplacko.tk
E-mail: gordonsweb@yahoo.com, marian.placko@zoznam.sk
Projekt: Ovládací panel cvičení
Verzia: 1.0.0
Dátum: 20. 01. 2005
Autor: © Marián Plačko
Poznámka: Centrálne ovládanie cvičení

Mikrokontrolér: AT90S8535
Hodinová frekvencia: 8,000000 MHz
Pamäťový model: Small
Vnútorná SRAM: 512 B
Externá SRAM: 0 B
Dátový zásobník: 128 B
*****************************************************************

Použitý LCD: MC16021E8-SYL, 2x16 znakov
Prepojenie MCU s LCD:
  [PINS on LCD]         [PINS on MCU AT90S8535, PORTB]
  01 GND    -           11 GND
  02 VCC    -           10 VCC
  03 Vee    -           LCD, kontrast, napätie: 0...1V
  04 RS     -           40 PB0
  05 RD     -           39 PB1
  06 EN     -           38 PB2
  07 DB0    -           11 GND
  08 DB1    -           11 GND
  09 DB2    -           11 GND
  10 DB3    -           11 GND
  11 DB0    -           36 PB4
  12 DB1    -           35 PB5
  13 DB6    -           34 PB6
  14 DB7    -           33 PB7
*****************************************************************/


// ***** Globálna inicializácia - BEGIN *****

#include <90s8535.h>
#include <delay.h>

// KeyPad
// Deklarácia globálneho poľa
char global_array [4] = {' ', ' ', ' ', ' '};  // Štvorprvkové pole,
inicializované na ' '

// LCD, PORTB
#asm
  .equ __lcd_port = 0x18
#endasm
#include <lcd.h>

// ***** Globálna inicializácia - END *****
```

```
// ***** LCD - BEGIN *****

// Inicializácia LCD
void lcd_initialization ()
{
  // Deklarácia privátnych premenných
  unsigned char col;  // StĺpceÍ
  unsigned char row;  // Riadky

  #asm ("cli");  // Zakáže všetky prerušenia

  // Inicializácia LCD (+ inicializácia LCD pre cvičenie 2, príklad 3
      - Running Logo)
  lcd_init (16);

// Zobrazenie Loga
  lcd_gotoxy (1, 0);
  lcd_putsf ("Marian  Placko");
  lcd_gotoxy (2, 1);
  lcd_putsf ("alias GORDON");
  delay_ms (1000);

  // Skrytie znakov na LCD
  for (col = 0; col < 16; col ++)
  {
    for (row = 0; row < 2; row ++)
    {
      lcd_gotoxy (col, row);
      lcd_putchar (0xFF);
    }
  }
  delay_ms (1000);

  // Postupné odkrývanie znakov na LCD
  for (col = 0; col < 16; col ++)
  {
    for (row = 0; row < 2; row ++)
    {
      lcd_gotoxy (col, row);
      lcd_putchar (0xFE);
      delay_ms (100);
    }
  }

  // Výpis hlavičky na LCD
  lcd_clear ();
  lcd_putsf ("Ovladaci panel");
  lcd_gotoxy (0, 1);
  lcd_putsf ("Zadajte cv:");
}
```

```
/*// Preblikávanie kurzora
void lcd_cursor ()
{
  #asm ("cli");  // Zakáže všetky prerušenia

  lcd_gotoxy (11,1);
  lcd_putsf ("_");
  delay_ms (500);
  lcd_gotoxy (11,1);
  lcd_putsf (" ");
  delay_ms (500);

  #asm ("sei");  // Povolenie všetkých prerušení
}*/

// ***** LCD - END *****



// ***** KeyPad - BEGIN *****

// Inicializácia  klávesnice
void keypad_initialization ()
{
  DDRC = 0xFF;   // Vstup z klávesnice
  TCCR0 = 1;     // Konštanty pre inicializáciu čítača 0
  TCNT0 = 0;
  TIMSK = 1;

  #asm ("sei");  // Povolenie všetkých prerušení
}


// Cvičenie 1, príklad 4 - Night Rider
/*#include <90s8535.h>
   #include <delay.h> */

void night_rider ()
{
  unsigned char aLED [8] = {0xFE, 0xFD, 0xFB, 0xF7, 0xEF, 0xDF, 0xBF,
                            0x7F};
  char c;  // Counter (počítadlo)

  DDRA = 0xFF;  // Inicializácia PORTA, výstup  na LED

  #asm ("cli");  // Zakáže všetky prerušenia

  while (1)
  {
    for (c = 0; c < 8; ++c)
    {
      PORTA = aLED [c];
      delay_ms (50);
    }
    for (c = 6; c > 0; --c)
    {
      PORTA = aLED [c];
      delay_ms (50);
    }
  }
}
```

```
// Cvičenie 2, príklad 3 - Running Logo
/* #include <90s8535.h>
    #include <delay.h>
    #asm
      .equ __lcd_port = 0x18;  PORTB
    #endasm
    #include <lcd.h> */

void running_logo ()
{
  unsigned char c;  // Counter (počítadlo)

  while (1)
  {
    // lcd_init (16);   // (Inicializácia pre LCD je použitá v "LCD
BEGIN")

    for (c = 0; c < 29; c ++)
    {
      if (c == 29) c = 0;      // Reset, návrat na začiatok

      lcd_gotoxy (c, 0);
      lcd_putsf ("o");
      lcd_gotoxy (c - 1, 0);  // Zmaže predchádzajúce písmená
      lcd_putsf ("k");
      lcd_gotoxy (c - 2, 0);
      lcd_putsf ("c");
      lcd_gotoxy (c - 3, 0);
      lcd_putsf ("a");
      lcd_gotoxy (c - 4, 0);
      lcd_putsf ("l");
      lcd_gotoxy (c - 5, 0);
      lcd_putsf ("P");
      lcd_gotoxy (c - 6, 0);
      lcd_putsf (" ");
      lcd_gotoxy (c - 7, 0);
      lcd_putsf ("n");
      lcd_gotoxy (c - 8, 0);
      lcd_putsf ("a");
      lcd_gotoxy (c - 9, 0);
      lcd_putsf ("i");
      lcd_gotoxy (c - 10, 0);
      lcd_putsf ("r");
      lcd_gotoxy (c - 11, 0);
      lcd_putsf ("a");
      lcd_gotoxy (c - 12, 0);
      lcd_putsf ("M");
      lcd_gotoxy (c - 13, 0);
      lcd_putsf (" ");
      lcd_gotoxy (0, 1);
      lcd_putsf (" ");

      delay_ms (250);
    }
  };
}
```

```
// Cvičenie 3, príklad 2 - Ascii Code
/* #include <90s8535.h> */
#include <stdio.h>

void ascii_code ()
{
  unsigned char c;

  // Inicializácia UART-u
  UCR = 0x08;
  UBRR = 0x33;

  printf ("Testing...");
  printf ("\n");
  c = '0';

  while (1)
  {
    putchar (c);
    c ++;
    if (c > 127)
    {
      c = '0';
      printf ("\n");
    }
  };
}


// Cvičenie 4, príklad 5 - Button - 7 Segment LED
/* #include <90s8535.h> */

void button_7segled ()
{
  unsigned char c;  // Counter (počítadlo)

  DDRA = 0xFF;   // Výstup na LED
  PORTA = 0xFF;
  PORTC = 0xFF;  // Vstup z tlačidiel
  c = 0;

  while (1)
  {
    if (PINC.7 == 0) c = 0x02;    // 0
    if (PINC.6 == 0) c = 0x3E;    // 1
    if (PINC.5 == 0) c = 0x84;    // 2
    if (PINC.4 == 0) c = 0x0C;    // 3
    if (PINC.3 == 0) c = 0x78;    // 4
    if (PINC.2 == 0) c = 0x48;    // 5
    if (PINC.1 == 0) c = 0x40;    // 6
    if (PINC.0 == 0) c = 0x1E;    // 7

    if (c != 0) PORTA = c;
  };
}
```

```
// Cvičenie 8, príklad 1 - MicroWire (TLC 549, meranie U, rozsah 0-5V)
/* TLC549 pripojený na  PORTD
    CS pripojený k PD3, D0 k PD4 , CLK k PD5
    Výstup na PORTA – LED diódy
    #include <90s8535.h>
    #include <delay.h> */
#define cs PORTD.3
#define clk PORTD.5

unsigned char readtlc()
{
  unsigned char napetie = 0, Ux = 128, loopi = 8;
  clk = 0;
  cs = 0;
  delay_us (2);
  while (loopi --)
  {
    if (PIND.4 == 1)  // Čítanie dát D0
    {
      napetie = napetie + Ux;
    }
    Ux = Ux / 2;
    delay_us (2);
    clk = 1; delay_us (2);  // Hodinový impulz
    clk = 0; delay_us (2);
  }
  cs = 1;
  return napetie;
}

void microwire_tlc549 ()
{
  // PORTA = 0x00;
  DDRA = 0xFF;
  // PORTD = 0x00;
  DDRD = 0x28;  // Pin0 = výstup, pin1=  vstup, pin2 = výstup

  while (1)
  {
    PORTA = readtlc ();  // Zobrazenie výsledku na PORTA – LED diódy
    delay_ms (100);
  }
}
```

```
// Výber cvičenia / príkladu z klávesnice
void keypad_cut (char key)
{
  // Zadanie 4 platných kombinácii cvičení resp. jednotlivé príklady k
      cvičeniam a ich zobrazenie na LCD
  if (global_array [0] ==  ' ') {global_array [0] = key; lcd_gotoxy
(12, 1); lcd_putchar (key);}
  else if (global_array [1] ==  ' ') {global_array [1] = key;
lcd_gotoxy (13, 1); lcd_putchar (key);}
  else if (global_array [2] ==  ' ') {global_array [2] = key;
lcd_gotoxy (14, 1); lcd_putchar (key);}
  else if (global_array [3] ==  ' ') {global_array [3] = key;
lcd_gotoxy (15, 1); lcd_putchar (key);}

  // Spustenie jednotlivých cvičení / príkladov
  // Cvičenie 1, príklad 4 - Night Rider
  if (global_array [0] == '1' & global_array [1] == '*' & global_array
[2] == '4' & global_array [3] == '#')
  {
    lcd_clear (); lcd_putsf ("Run...");
    night_rider ();  // Cvičenie 1, príklad 4
  }
  // Cvičenie 2, príklad 3 - Running Logo
  else if (global_array [0] == '2' & global_array [1] == '*' &
global_array [2] == '3' & global_array [3] == '#')
  {
    lcd_clear (); lcd_putsf ("Run..."); delay_ms (500);
    running_logo ();  // Cvičenie 2, príklad 3
  }
  // Cvičenie 3, príklad 2 - Ascii Code
  else if (global_array [0] == '3' & global_array [1] == '*' &
global_array [2] == '2' & global_array [3] == '#')
  {
    lcd_clear (); lcd_putsf ("Run...");
    ascii_code ();  // Cvičenie 3, príklad 2
  }
  // Cvičenie 5, príklad 5 - Button - 7 Segment LED
  else if (global_array [0] == '4' & global_array [1] == '*' &
global_array [2] == '5' & global_array [3] == '#')
  {
    lcd_clear (); lcd_putsf ("Run...");
    button_7segled ();  // Cvičenie 4, príklad 5
  }
  // Cvičenie 8, príklad 1 - MicroWire (TLC 549, meranie U, rozsah 0-
5V)
  else if (global_array [0] == '8' & global_array [1] == '*' &
global_array [2] == '1' & global_array [3] == '#')
  {
    lcd_clear (); lcd_putsf ("Run..."); delay_ms (500);
    microwire_tlc549 ();  // Cvičenie 8, príklad 1
  }
  // Cvičenie / príklad neexistuje, po zadaní štvrtého znaku
  else if (global_array [3] != ' ') {lcd_clear (); lcd_putsf
("Error");}
}
```

```
// Vyhodnotenie stlačenia tlačidiel
void keypad_keycode ()
{
  // Deklarácia privátnej premennej
  #define KEYIN PINC  // Vstupné riadky

  // Samotné vyhodnotenie
  KEYIN = PORTC;

  switch (KEYIN)
  {
    case 0xD7 :  // Tlačidlo "1"
      keypad_cut ('1');
      while (KEYIN == 0xD7)  // Uvoľnenie tlačidla
      {delay_ms (20);}  // Ošetrenie tlačidlo na zákmity
    break;
    case 0xB7 :  // Tlačidlo "2"
      keypad_cut ('2');
      while (KEYIN == 0xB7)  // Uvoľnenie tlačidla
      {delay_ms (20);}  // Ošetrenie tlačidla na zákmity
    break;
    case 0xE7 :  // Tlačidlo "3"
      keypad_cut ('3');
      while (KEYIN == 0xE7)  // Uvoľnenie tlačidla
      {delay_ms (20);}  // Ošetrenie tlačidla na zákmity
    break;
    case 0xDB :  // Tlačidlo "4"
      keypad_cut ('4');
      while (KEYIN == 0xDB)  // Uvoľnenie tlačidla
      {delay_ms (20);}  // Ošetrenie tlačidla na zákmity
    break;
    case 0xBB :  // Tlačidlo "5"
      keypad_cut ('5');
      while (KEYIN == 0xBB)  // Uvoľnenie tlačidla
      {delay_ms (20);}  // Ošetrenie tlačidla na zákmity
    break;
    case 0xEB :  // Tlačidlo "6"
      keypad_cut ('6');
      while (KEYIN == 0xEB)  // Uvoľnenie tlačidla
      {delay_ms (20);}  // Ošetrenie tlačidla na zákmity
    break;
    case 0xDD :  // Tlačidlo "7"
      keypad_cut ('7');
      while (KEYIN == 0xDD)  // Uvoľnenie tlačidla
      {delay_ms (20);}  // Ošetrenie tlačidla na zákmity
    break;
    case 0xBD :  // Tlačidlo "8"
      keypad_cut ('8');
      while (KEYIN == 0xBD)  // Uvoľnenie tlačidla
      {delay_ms (20);}  // Ošetrenie tlačidla na zákmity
    break;
    case 0xED :  // Tlačidlo "9"
      keypad_cut ('9');
      while (KEYIN == 0xED)  // Uvoľnenie tlačidla
      {delay_ms (20);}  // Ošetrenie tlačidlo na zákmity
    break;
    case 0xDE :  // Tlačidlo "*"
      keypad_cut ('*');
      while (KEYIN == 0xDE)  // Uvoľnenie tlačidla
      {delay_ms (20);}  // Ošetrenie tlačidla na zákmity
    break;
```

```
    case 0xBE :  // Tlačidlo "0"
      keypad_cut ('0');
      while (KEYIN == 0xBE)  // Uvoľnenie tlačidla
      {delay_ms (20);}  // Ošetrenie tlačidla na zákmity
    break;
    case 0xEE :  // Tlačidlo "#"
      keypad_cut ('#');
      while (KEYIN == 0xEE)  // Uvoľnenie tlačidla
      {delay_ms (20);}  // Ošetrenie tlačidla na zákmity
  }
}

// Prerušenie (stĺpce)
interrupt [TIM0_OVF] void timer0_int (void)
{
  PORTC = 0b11011111;  // Stĺpec: 1, tlačidlá: 1, 4, 7, *
  keypad_keycode ();

  PORTC = 0b10111111;  // Stĺpec: 2, tlačidlá: 2, 5, 8, 0
  keypad_keycode ();

  PORTC = 0b11101111;  // Stĺpec: 2, tlačidlá: 2, 5, 8, 0
  keypad_keycode ();
}

// ***** KeyPad - END *****



// ***** Program - BEGIN *****

void main ()
{
  lcd_initialization ();
  keypad_initialization ();
  while (1)
  {
    // lcd_cursor ();
  }
}

// ***** Program - END *****

/*****************************************************************/
```

CD PRÍLOHY

---

2. Prehľad "rodín" mikrokontrolérov
3. Bloková štruktúra AT90S8535
4. Rozloženie vývodov na AT90S8535
5. Inštrukčná sada pre mikrokontroléry ATMEL
6. Kompilátory a vývojové prostredia pre mikrokontroléry ATMEL AVR
7. Prehľad a opis elektronických súčiastok
8. Inteligentné displeje LCD
9. RS232

(Appendix for the Graduation Theses)

PRÍLOHA 2 – PREHĽAD "RODÍN" MIKROKONTROLÉROV

Typ	Hod. MHz	ROM	RAM bytov	V/V 8 bit	A/D bit	Čas./ čítač	Čas. WD	Prer. úrov.	Seriál	Za/Ko PWM	Puzdro
INTEL											
80C31	33	0	128	4	-	2	-	5/2	UART	-	DIP40, PLCC44
80C51	33	4KB	128	4	-	2	-	5/2	UART	-	PLCC44
87C51	33	E4KB	128	4	-	2	-	5/2	UART	-	PLCC44
8xC51FA	12/16	8KB	256	4	Cout	3	0+1	7/4	UART	5/5/5	PLCC44
8xC51FB	12/16	16KB	256	4	Cout	3	0+1	7/4	UART	5/5/5	PLCC44
8xC51FC	12/16	32KB	256	4	Cout	3	0+1	7/4	UART	5/5/5	PLCC44
80C251SB	12/16	16KB	1KB	4	Cout	4	0+1	7/4	UART	5/5/5	PLCC44
PHILIPS											
Väčšina uvedených verzií je v prevedení bez ROM (80), v prevedení OTP (83), v prevedení EPROM (87)											
87C52	24	E8KB	256	4	-	3	-		UART	1/0/0	PLCC44, PQFP44
87C54	24	E16KB	256	4	-	3	-		UART	1/0/0	
87C58	24	E32KB	256	4	-	3	-		UART	1/0/0	
87C550	16	E4KB	128	4	8/8	2	1	6/2	UART	-	PLCC44
87C524	12/20	E16KB	512	4	-	3	1	6/2	UART, I2C	-	PLCC44
83C552	16/24	8KB	256	5+1	8/10	3	0+1	15/2	UART, I2C	4/8/2	PLCC68
89C558	16	F32KB	1KB	5+1	8/10	3	0+1	15/2	UART, I2C	4/8/2	PQFP80
87C575	16	E8KB	256	4	4 kom.	3	1	7/2	UART	5-PCA	PLCC44
87C576	16	E8KB	256	4	8/10	3	1		UART, I2C	PCA	PLCC44
87C654	16/24	E16KB	256	4	-	2	-	6/2	UART, I2C	-	PLCC44
87C751	16	E2KB	64	2+3b	-	1	-	5/2	I2C	-	PLCC28
87C752	16	E2KB	64	2+5b	5/8	1	-	7/2	I2C	0/0/1	PLCC28
83C852	16	6KB, EE2KB	256	4	-	2	-	5/2	UART	-	PLCC44
SIEMENS											
SAB80C515 SAB80C535	12-16	8 -	256	6+1	8/8	3	1+1	12/4	UART	4/4/0	PLCC68
SAB80C517 SAB80C537	12/16	8 -	256	7+1,5	12/8	4	2+2	14/4	2xUART	5/21/5	PLCC84 PQFP100
DALLAS											
DS80C320	25	-	256	4	-	3	1	13/2	2xUART	1/0/0	PLCC44
DS80C320	25	E16KB	256 +1KB	4	-	3	1	13/2	2xUART	1/0/0	PLCC44, TQFP44
DS80C320	25	E16KB	256 +1KB	4	RTC	3	1	14/2	2xUART	1/0/0	PLCC44, TQFP44

Tab 1. Prehľad mikrokontrolérov INTEL, PHILIPS, SIEMENS a DALLAS

Typ	RAM bytov	Flash EPROM KB	EEPROM KB	Frekv. MHz	IO linky	Čas.	Prer.	Anal. kom.	SPI	IDLE, PD	Dual Data Pointer	Watch Dog	Power Off Flag	CMOS
AT89C1051	64	1	-	0-24	15	2	6	ÁNO	-	ÁNO	-	-	-	ÁNO
AT89C2051	128	2	-	0-24	15	2	6	ÁNO	-	ÁNO	-	-	-	ÁNO
AT89C4051	128	4	-	0-24	15	2	6	ÁNO	-	ÁNO	-	-	-	ÁNO
AT89C51	128	4	-	0-24	32	2	6	-	-	ÁNO	-	-	-	ÁNO
AT89C52	256	8	-	0-24	32	3	9	-	-	ÁNO	-	-	-	ÁNO
AT89S53	256	12	-	0-24	32	3	8	-	ÁNO	ÁNO	ÁNO	ÁNO	ÁNO	ÁNO
AT89C55	256	20	-	0-33	32	3	6	-	-	ÁNO	-	-	-	ÁNO
AT89S8252	256	8	2	0-24	32	3	9	-	ÁNO	ÁNO	ÁNO	ÁNO	ÁNO	ÁNO
AT89S4D12	256	128	-	12	5	-	-	-	ÁNO	-	ÁNO	-	-	ÁNO

Tab 2. Prehľad a vlastnosti mikrokontrolérov ATMEL s jadrom 8051

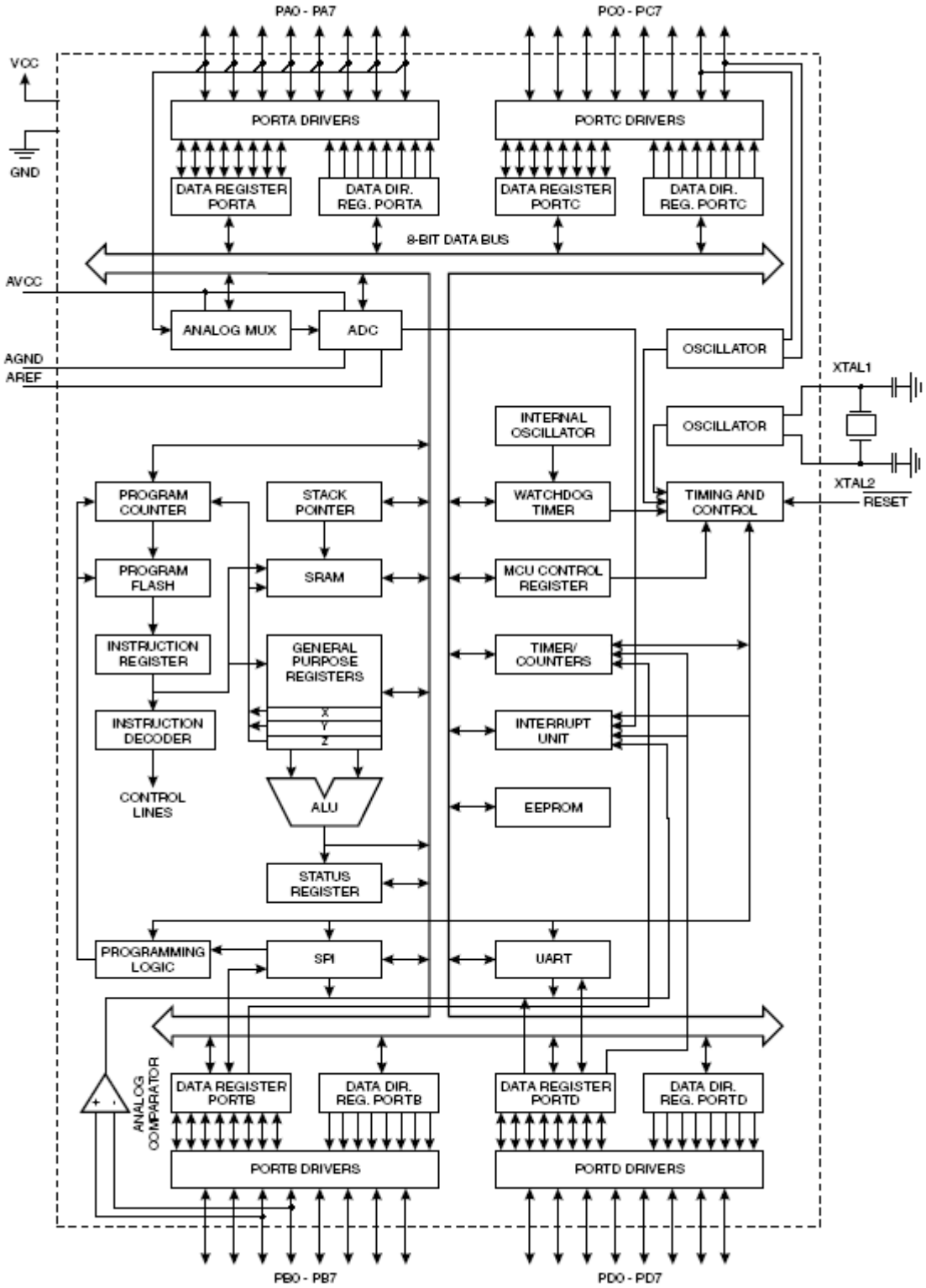
Typ	1200	2313	2343	4433	8515	8535
Max. frekv. CPU v [MHz]	4 popr. 12	4 popr. 10	1, 4, 10	4 popr. 8	4 popr. 8	4 popr. 8
Poč. inštrukcií	89	118	118	118	118	118
Poč. obecných registrov	32	32	32	32	32	32
FLASH	1KB	2KB	2KB	4KB	8KB	8KB
SRAM	-	128B	128B	128B	512B	512B
EEPROM	64B	128B	128B	256B	512B	512B
Čítač/časovač0	8 bit	8 bit	8 bit	8 bit	8 bit	8 bit
Čítač/časovač1	-	16 bit	-	16 bit	16 bit	16 bit
Čítač/časovač2	-	-	-	-	-	8 bit
Watchdog	ÁNO	ÁNO	ÁNO	ÁNO	ÁNO	ÁNO
A/D prevodník	-	-	-	10 bit	-	10 bit
Analóg. komp.	ÁNO	ÁNO	ÁNO	ÁNO	ÁNO	ÁNO
UART	-	ÁNO	ÁNO	ÁNO	ÁNO	ÁNO
SPI	ÁNO	ÁNO	ÁNO	ÁNO	ÁNO	ÁNO
PORT A	-	-	-	-	ÁNO	ÁNO
PORT B	ÁNO	ÁNO	ÁNO	ÁNO	ÁNO	ÁNO
PORT C	-	-	-	ÁNO	ÁNO	ÁNO
PORT D	ÁNO	ÁNO	-	ÁNO	ÁNO	ÁNO

Tab 3. Prehľad a vlastnosti mikrokontrolérov ATMEL AVR − základná rada

Typ	Flash	EEPROM	Register	Napájacie U	Frekvencia
ATtiny11L	1KB	-	32	2.7 – 5.5 V	0 – 2 MHz
ATtiny11	1KB	-	32	4.0 – 5.5 V	0 – 6 MHz
ATtiny12V	1KB	64B	32	1.8 – 5.5 V	0 – 1.2 MHz
ATtiny12L	1KB	64B	32	2.7 – 5.5 V	0 – 4 MHz
ATtiny12	1KB	64B	32	4.0 – 5.5 V	0 – 8 MHz

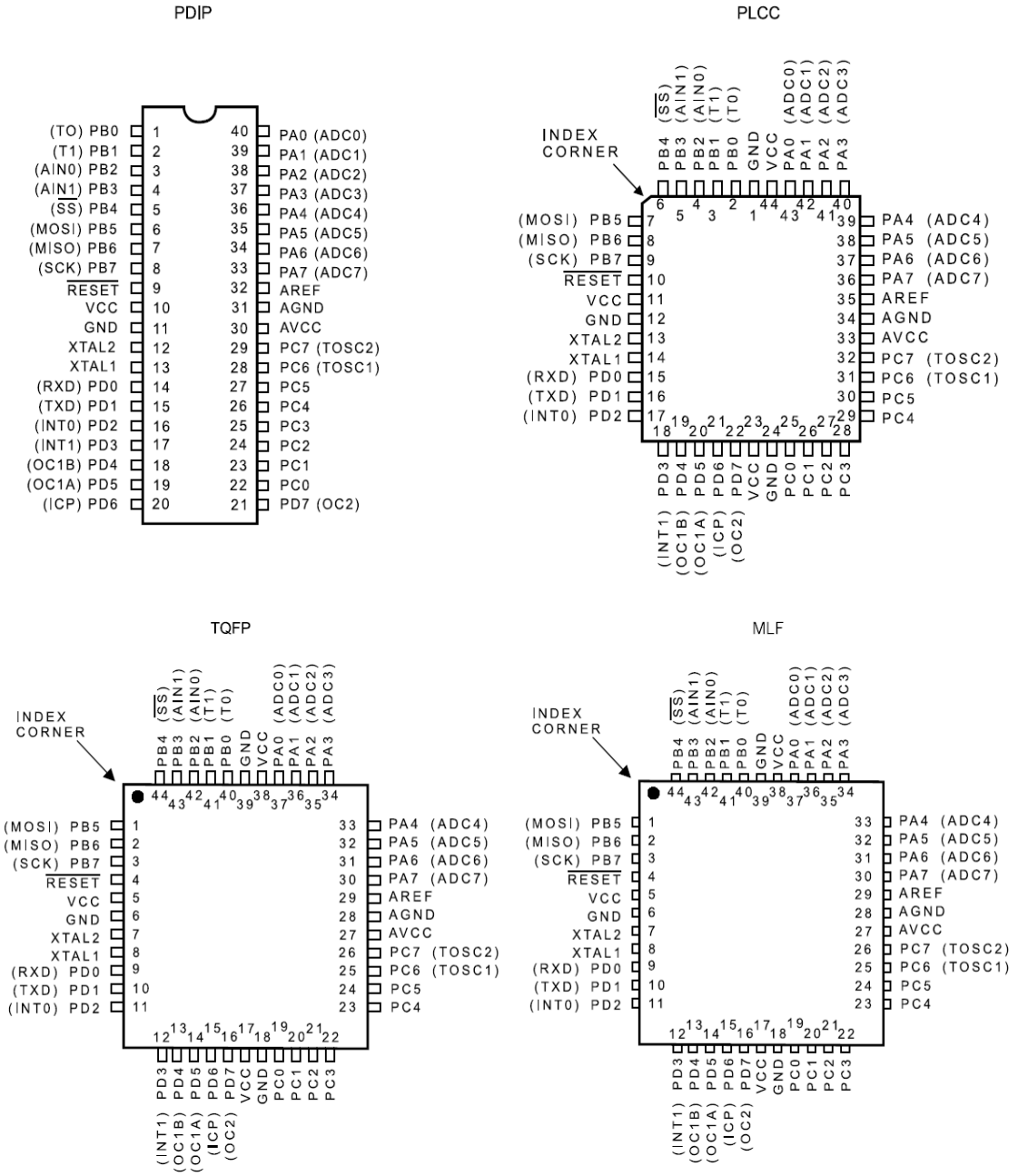
Tab 4. Prehľad a vlastnosti mikrokontrolérov ATMEL AtTiny

PRÍLOHA 3 − BLOKOVÁ ŠTRUKTÚRA AT90S8535



Obr. 1 Bloková štruktúra AT90S8535. (www.atmel.com)

PRÍLOHA 4 – ROZLOŽENIE VÝVODOV NA AT90S8535



Obr. 1 Rozloženie vývodov na AT90S8535

PRÍLOHA 5 − INŠTRUKČNÁ SADA PRE MIKROKONTROLÉRY ATMEL

Tab 1. Zoznam inštrukčnej sady pre mikrokontroléry Atmel

Status Register (SREG):

SREG: Status register
C: Carry flag in status register
Z: Zero flag in status register
N: Negative flag in status register
V: Twos complement overflow indicator
S: N ? V, For signed tests
H: Half Carry flag in the status register
T: Transfer bit used by BLD and BST instructions
I: Global interrupt enable/disable flag

Registers and operands:

Rd: Destination (and source) register in the register file
Rr: Source register in the register file
R: Result after instruction is executed
K: Constant literal or byte data (8 bit)
k: Constant address data for program counter
b: Bit in the register file or I/O register (3 bit)
s: Bit in the status register (3 bit)
X,Y,Z: Indirect address register (X=R27:R26, Y=R29:R28 and Z=R31:R30)
P: I/O port address
q: Displacement for direct addressing (6 bit)

I/O Registers

RAMPX, RAMPY, RAMPZ:
Registers concatenated with the X, Y and Z registers enabling indirect addressing of the whole SRAM area on MCUs with more than 64K bytes SRAM.

Stack:

STACK:Stack for return address and pushed registers
SP: Stack Pointer to STACK
Flags:
?: Flag affected by instruction
0: Flag cleared by instruction
1: Flag set by instruction
-: Flag not affected by instruction

Tab 2. Aritmetické a logické inštrukcie

Názov	Operandy	Opis	Operácia	Príznaky	Takty
ADD	Rd, Rr	Add without Carry	Rd ← Rd + Rr	Z,C,N,V,H	1
ADC	Rd, Rr	Add with Carry	Rd ← Rd + Rr + C	Z,C,N,V,H	1
ADIW	Rd, K	Add Immediate to Word	Rd+1:Rd ← Rd+1:Rd + K	Z,C,N,V	2
SUB	Rd, Rr	Subtract without Carry	Rd ← Rd - Rr	Z,C,N,V,H	1
SUBI	Rd, K	Subtract Immediate	Rd ← Rd - K	Z,C,N,V,H	1
SBC	Rd, Rr	Subtract with Carry	Rd ← Rd - Rr - C	Z,C,N,V,H	1
SBCI	Rd, K	Subtract Immediate with Carry	Rd ← Rd - K - C	Z,C,N,V,H	1
SBIW	Rd, K	Subtract Immediate from Word	Rd+1:Rd ← Rd+1:Rd - K	Z,C,N,V	2
AND	Rd, Rr	Logical AND	Rd ← Rd · Rr	Z,N,V	1
ANDI	Rd, K	Logical AND with Immediate	Rd ← Rd · K	Z,N,V	1
OR	Rd, Rr	Logical OR	Rd ← Rd v Rr	Z,N,V	1
ORI	Rd, K	Logical OR with Immediate	Rd Rd v K	Z,N,V	1
EOR	Rd, Rr	Exclusive OR	Rd ← Rd ⊕ Rr	Z,N,V	1
COM	Rd	One's Complement	Rd ← $FF - Rd	Z,C,N,V	1
NEG	Rd	Two's Complement	Rd ← $00 - Rd	Z,C,N,V,H	1
SBR	Rd,K	Set Bit(s) in Register	Rd ← Rd v K	Z,N,V	1
CBR	Rd,K	Clear Bit(s) in Register	Rd ← Rd · ($FFh - K)	Z,N,V	1
INC	Rd	Increment	Rd ← Rd + 1	Z,N,V	1
DEC	Rd	Decrement	Rd ← Rd - 1	Z,N,V	1
TST	Rd	Test for Zero or Minus	Rd ← Rd · Rd	Z,N,V	1
CLR	Rd	Clear Register	Rd ← Rd ⊕ Rd	Z,N,V	1
SER	Rd	Set Register	Rd ← $FF	None	1
CP	Rd,Rr	Compare	Rd - Rr	Z,C,N,V,H	1
CPC	Rd,Rr	Compare with Carry	Rd - Rr - C	Z,C,N,V,H	1
CPI	Rd,K	Compare with Immediate	Rd - K	Z,C,N,V,H	1

Tab 3. Inštrukcie vetvenia

Názov	Operandy	Opis	Operácia	Príznaky	Takty
RJMP	k	Relative Jump	PC ← PC + k + 1	None	2
IJMP		Indirect Jump to (Z)	PC ← Z	None	2
JMP	k	Jump	PC ← k	None	3
RCALL	k	Relative Call Subroutine	PC ← PC + k + 1	None	3
ICALL		Indirect Call to (Z)	PC ← Z	None	3
CALL	k	Call Subroutine	PC ← k	None	4
RET		Subroutine Return	PC ← STACK	None	4
RETI		Interrupt Return	PC ← STACK	I	4
CPSE	Rd,Rr	Compare, Skip if Equal	if (Rd = Rr) PC ← PC + 2 or 3	None	1 / 2 / 3
SBRC	Rr, b	Skip if Bit in Register Cleared	if (Rr(b)=0) PC ← PC + 2 or 3	None	1 / 2 / 3
SBRS	Rr, b	Skip if Bit in Register Set	if (Rr(b)=1) PC ← PC + 2 or 3	None	1 / 2 / 3
SBIC	P, b	Skip if Bit in I/O Register Cleared	if(I/O(P,b)=0) PC ← PC + 2 or 3	None	1 / 2 / 3
SBIS	P, b	Skip if Bit in I/O Register Set	If(I/O(P,b)=1) PC ← PC + 2 or 3	None	1 / 2 / 3
BRBS	s, k	Branch if Status Flag Set	if (SREG(s) = 1) then PC ← PC+k + 1	None	1 / 2
BRBC	s, k	Branch if Status Flag Cleared	if (SREG(s) = 0) then PC ← PC+k + 1	None	1 / 2
BREQ	k	Branch if Equal	if (Z = 1) then PC ← PC + k + 1	None	1 / 2
BRNE	k	Branch if Not Equal	if (Z = 0) then PC ← PC + k + 1	None	1 / 2
BRCS	k	Branch if Carry Set	if (C = 1) then PC ← PC + k + 1	None	1 / 2
BRCC	k	Branch if Carry Cleared	if (C = 0) then PC ← PC + k + 1	None	1 / 2
BRSH	k	Branch if Same or Higher	if (C = 0) then PC PC + k + 1	None	1 / 2
BRLO	k	Branch if Lower	if (C = 1) then PC ← PC + k + 1	None	1 / 2
BRMI	k	Branch if Minus	if (N = 1) then PC ← PC + k + 1	None	1 / 2
BRPL	k	Branch if Plus	if (N = 0) then PC ← PC + k + 1	None	1 / 2

Názov	Operandy	Opis	Operácia	Príznaky	Takty
BRGE	k	Branch if Greater or Equal, Signed	if (N ⊕ V= 0) then PC ← PC+ k + 1	None	1 / 2
BRLT	k	Branch if Less Than, Signed	if (N ⊕ V= 1) then PC ← PC + k + 1	None	1 / 2
BRHS	k	Branch if Half Carry Flag Set	if (H = 1) then PC ← PC + k + 1	None	1 / 2
BRHC	k	Branch if Half Carry Flag Cleared	if (H = 0) then PC ← PC + k + 1	None	1 / 2
BRTS	k	Branch if T Flag Set	if (T = 1) then PC ← PC + k + 1	None	1 / 2
BRTC	k	Branch if T Flag Cleared	if (T = 0) then PC ← PC + k + 1	None	1 / 2
BRVS	k	Branch if Overflow Flag is Set	if (V = 1) then PC ← PC + k + 1	None	1 / 2
BRVC	k	Branch if Overflow Flag is Cleared	if (V = 0) then PC ← PC + k + 1	None	1 / 2
BRIE	k	Branch if Interrupt Enabled	if (I = 1) then PC ← PC + k + 1	None	1 / 2
BRID	k	Branch if Interrupt Disabled	if (I = 0) then PC ← PC + k + 1	None	1 / 2

Tab 4. Inštrukcie pre prenos dát

Názov	Operandy	Opis	Operácia	Príznaky	Takty
MOV	Rd, Rr	Copy Register	Rd ← Rr	None	1
LDI	Rd, K	Load Immediate	Rd ← K	None	1
LDS	Rd, k	Load Direct from SRAM	Rd ← (k)	None	3
LD	Rd, X	Load Indirect	Rd ← (X)	None	2
LD	Rd, X+	Load Indirect and Post-Increment	Rd ← (X), X ← X + 1	None	2
LD	Rd, -X	Load Indirect and Pre-Decrement	X ← X - 1, Rd (X)	None	2
LD	Rd, Y	Load Indirect	Rd ← (Y)	None	2
LD	Rd, Y+	Load Indirect and Post-Increment	Rd ← (Y), Y ← Y + 1	None	2
LD	Rd, -Y	Load Indirect and Pre-Decrement	Y ← Y - 1, Rd ← (Y)	None	2
LDD	Rd,Y+q	Load Indirect with Displacement	Rd ← (Y + q)	None	2
LD	Rd, Z	Load Indirect	Rd ← (Z)	None	2

Názov	Operandy	Opis	Operácia	Príznaky	Takty
LD	Rd, Z+	Load Indirect and Post-Increment	Rd ← (Z), Z ← Z+1	None	2
LD	Rd, -Z	Load Indirect and Pre-Decrement	Z ← Z - 1, Rd ← (Z)	None	2
LDD	Rd, Z+q	Load Indirect with Displacement	Rd ← (Z + q)	None	2
STS	k, Rr	Store Direct to SRAM	Rd ← (k)	None	3
ST	X, Rr	Store Indirect	(X) ← Rr	None	2
ST	X+, Rr	Store Indirect and Post-Increment	(X) ← Rr, X ← X + 1	None	2
ST	-X, Rr	Store Indirect and Pre-Decrement	X ← X - 1, (X) Rr	None	2
ST	Y, Rr	Store Indirect	(Y) ← Rr	None	2
ST	Y+, Rr	Store Indirect and Post-Increment	(Y) ← Rr, Y ← Y + 1	None	2
ST	-Y, Rr	Store Indirect and Pre-Decrement	Y ← Y - 1, (Y) ← Rr	None	2
STD	Y+q,Rr	Store Indirect with Displacement	(Y + q) ← Rr	None	2
ST	Z, Rr	Store Indirect	(Z) ← Rr	None	2
ST	Z+, Rr	Store Indirect and Post-Increment	(Z) ← Rr, Z ← Z + 1	None	2
ST	-Z, Rr	Store Indirect and Pre-Decrement	Z ← Z - 1, (Z) ← Rr	None	2
STD	Z+q,Rr	Store Indirect with Displacement	(Z + q) ← Rr	None	2
LPM		Load Program Memory	R0 ← (Z)	None	3
IN	Rd, P	In Port	Rd ← P	None	1
OUT	P, Rr	Out Port	P ← Rr	None	1
PUSH	Rr	Push Register on Stack	STACK ← Rr	None	2
POP	Rd	Pop Register from Stack	Rd ← STACK	None	2

Tab 5. Bitové inštrukcie

Názov	Operandy	Opis	Operácia	Príznaky	Takty
LSL	Rd	Logical Shift Left	Rd(n+1) ← Rd(n),Rd(0) ← 0,C ← Rd(7)	Z,C,N,V,H	1
LSR	Rd	Logical Shift Right	Rd(n) ← Rd(n+1),Rd(7) ← 0,C ← Rd(0)	Z,C,N,V	1
ROL	Rd	Rotate Left Through Carry	Rd(0) ← C,Rd(n+1) ← Rd(n),C ← Rd(7)	Z,C,N,V,H	1
ROR	Rd	Rotate Right Through Carry	Rd(7) ← C,Rd(n) ← Rd(n+1),C ← Rd(0)	Z,C,N,V	1
ASR	Rd	Arithmetic Shift Right	Rd(n) ← Rd(n+1), n=0..6	Z,C,N,V	1
SWAP	Rd	Swap Nibbles	Rd(3..0) ↔ Rd(7..4)	None	1
BSET	s	Flag Set	SREG(s) ← 1	SREG(s)	1
BCLR	s	Flag Clear	SREG(s) ← 0	SREG(s)	1
SBI	P, b	Set Bit in I/O Register	I/O(P, b) ← 1	None	2
CBI	P, b	Clear Bit in I/O Register	I/O(P, b) ← 0	None	2
BST	Rr, b	Bit Store from Register to T	T ← Rr(b)	T	1
BLD	Rd, b	Bit load from T to Register	Rd(b) ← T	None	1
SEC		Set Carry	C ← 1	C	1
CLC		Clear Carry	C ← 0	C	1
SEN		Set Negative Flag	N ← 1	N	1
CLN		Clear Negative Flag	N ← 0	N	1
SEZ		Set Zero Flag	Z ← 1	Z	1
CLZ		Clear Zero Flag	Z ← 0	Z	1
SEI		Global Interrupt Enable	I ← 1	I	1
CLI		Global Interrupt Disable	I ← 0	I	1
SES		Set Signed Test Flag	S ← 1	S	1
CLS		Clear Signed Test Flag	S ← 0	S	1
SEV		Set Two's Complement Overflow	V ← 1	V	1

Názov	Operandy	Opis	Operácia	Príznaky	Takty
CLV		Clear Two's Complement Overflow	V ← 0	V	1
SET		Set T in SREG	T ← 1	T	1
CLT		Clear T in SREG	T ← 0	T	1
SEH		Set Half Carry Flag in SREG	H ← 1	H	1
CLH		Clear Half Carry Flag in SREG	H ← 0	H	1
NOP		No Operation	None	1	
SLEEP		Sleep	(see specific descr. for Sleep)	None	1
WDR		Watchdog	Reset (see specific descr. for WDR)	None	1

PRÍLOHA 6 − KOMPILÁTORY A VÝVOJOVÉ PROSTREDIA PRE MIKROKONTROLÉRY ATMEL

Zoznam kompilátorov jazyka C:

- Dunfield Development Systems - http://www.dunfield.com/ ,
- HI-TECH Software - http://www.htsoft.com/ ,
- IAR Systems - http://www.iar.com/ ,
- Keil Software, Inc. - http://www.keil.com/ ,
- Franklin Software - http://www.fsinc.com/ ,
- Altium Limited - http://www.tasking.com/ ,
- Avocet Systems, Inc. - http://www.avocetsystems.com/ ,
- Small Device C Compiler - http://sdcc.sourceforge.net/ ,
- Wickenhäuser Elektrotechnik mC51 - http://www.wickenhaeuser.com/ ,
- Raisonance - http://%20www.raisonance.com/ ,
- Crossware Products - http://www.crossware.com/ ,
- Rigel Corporation Reads51 - http://www.rigelcorp.com/ ,
- a ďalšie.

Podstatné rozdiely medzi kompilátormi

Objektívne zhodnotiť vlastnosti všetkých kompilátorov je pomaly nadľudská a časovo náročná úloha. Preto by sme sa pokúsili s Vami podeliť o pár osobných skúseností, ktoré sme nadobudli u kompilátorov, čo sme mali možnosť vyskúšať (Keil, SDCC, Franklin, Hi-Tech, Reads51, µC51).

- vždy sa pozrieť na dátum poslednej aktualizácie daného kompilátora. Používať pokiaľ možno poslednú verziu programu (SDCC). Je to nepochopiteľné, že niekto je schopný predávať kompilátor, ktorý už niekoľko rokov nevyvíja (Franklin Software),
- aké klony x51 sú skutočne podporované, nejedná sa iba o základný rad 8051/52? (SDCC). Je možné zvoliť preklad aplikácie pre daný typ procesora, rozšírenou inštrukčnou sadou?, … (Keil),
- napísať si triviálnu aplikáciu a tu skúsiť preložiť pod viacerými prekladačmi. Pozrieť sa, či je pre Váš procesor k dispozícií hlavičkový súbor. V akom formáte sú zapisované hlavičkové súbory, je možné použiť naše stávajúce hlavičkové súbory i inde?
- jedná sa o zavedený kompilátor (Keil, Hi-Tech, …), alebo sa niekto snaží vytvoriť svoju vlastnú variantu C (µC51)?
- je to plnohodnotné C alebo kompilátor z radu „small C“ (Reads51), ktorý je skôr vhodný na hranie než na praktické aplikácie?
- ako pracuje daný kompilátor so zásobníkom? Nemá kompilátor tendenciu obsadzovať pamäť a neuvoľňovať ju (SDCC)?
- je vývojové prostredie skutočne Win32 aplikáciou, nespúšťa sa náhodou iba v DOS-ovom okne (Hi-Tech)?
- má výrobca skúsenosti i s kompilátormi pre iné typy procesorov než x51?

- aký formát ukladania dát používa daný kompilátor big/little endian (Keil/SDCC)?

Obecne je možné povedať, že väčšina kompilátorov nemá problémy, pokiaľ programovaná aplikácia vystačí s prostriedkami dostupnými priamo na čipe (model small). Skutočným testom, kde je možné vidieť čo kompilátor dokáže, je približne aplikácia okolo 1000 a viac riadkov zdrojového kódu. Najlepšie kompilovaná v modeli large s externou pamäťou XRAM a základnými periférіami.

Bohužiaľ takto zložitú aplikáciu, ako celok nie je vôbec jednoduché previesť z jedného kompilátoru do druhého. Obzvlášť pokiaľ tak robíme prvýkrát, keď je vyžadovaná nutná dávka entuziazmu. Dôvod je jednoduchý. Veľa vecí sa dá zapísať v jazyku C rôznymi spôsobmi, ale každý kompilátor si zdrojový kód interpretuje trochu inak. Pokiaľ tieto skúsenosti vývojár raz nadobudne, potom mu nerobí problémy vyvíjať aplikácie pre viac typov kompilátorov.

SDCC vyvíjané skupinou nadšencov si v porovnaní s niektorými komerčnými produktmi vedie vcelku obstojne. Chýba mnoho vymožeností, ktoré sú dané absenciou IDE a s tým spojený komfort pri vývoji. Rozšírená syntax jazyka C používaná u mikrokontrolérov radu x51 je u SDCC a Keil C51 zhodná. To umožňuje písať programy, ktoré je možné jednoducho preložiť v obidvoch kompilátoroch.

Keil C51 je odbornou verejnosťou považovaný za jeden z najlepších kompilátorov pre x51. Je to pravda a na celom produkte je vidieť niekoľko rokov intenzívneho vývoja, ktorý má za sebou. Pokiaľ vážne uvažujeme o nákupe komerčného kompilátora, tak by sme pri výbere nemali opomenúť Keil C51.

Porovnanie práce 2 IDE

1. **µVision2 IDE − Integrované vývojové prostredie od firmy Keil Software, Inc**

Firma Keil Software bola založená v roku 1986, keď sa zaoberala predajom vývojových nástrojov ponúkaných niekoľkými výrobcami. Zanedlho začala firma vyvíjať vlastné vývojové nástroje a dnes ich softvér patrí k špičke na trhu a podľa neho sa hodnotia vývojové programy ostatných firiem. Tento článok má za úlohu podrobnejšie popísať ich vývojový softvér µVision2.

Prečo sú kompilátory (a celkovo softvérové nástroje) firmy Keil tak kvalitné a celosvetovo uznávanou špičkou? Jeden z dôvodov je samozrejme ten, že sa zaujímajú vývojom už takmer 20 rokov. Druhý dôvod súvisiaci so špecializáciou na úzku škálu obvodov. µVision2 IDE je "len" pre tie typy mikrokontrolérov uvedené v nasledujúcom zozname a pre tieto obvody sa softvéroví inžinieri od Keil-u snažia vytvoriť čo najlepšiu technickú a programovú podporu.

Podporované obvody z rodiny:

- 8051 *****
- 80251
- C16x/ST10
- ARM7 ******

***** Pre mikrokontroléry s jadrom 51 sa jedná o skutočne najlepší nástroj, ktorý je možné na trhu získať, lebo má ako jediný zvládnuté odswapovavanie zásobníka a ktorý podporuje i super výkonné typy s týmto jadrom (Cygnal, Dallas...).

****** Podpora pre tieto obvody sa teraz pripravuje. Do µVision2 IDE by mala byť zakomponovaná behom niekoľkých mesiacov.

Popis vývojového prostredia

Vývojové nástroje obslužného softvéru pre mikrokontroléry z týchto rodín sú takmer všetky implementované do prostredia µVision2 IDE, tzn. kompilátor, assembler, RTOS, debugger, projektový manažér, editor zdrojového kódu a nástroje pre kompletnú simuláciu. Pre inštaláciu je potreba minimálne 16 MB RAM, 30 MB voľného miesta na disku a systém Winows98/NT4.0/2000/XP.

Vlastnosti IDE:

- podpora všetkých derivátov 8051,
- podpora 32-bitového IEEE floating-point (pohyblivá čiarka),
- možnosť bitovej adresácie objektov,
- použitie inštrukcií ACALL a AJMP,
- prístup do SFR registrov na úrovni C,
- podpora stránkovania pamäti,
- priama podpora prerušenia na úrovni C.

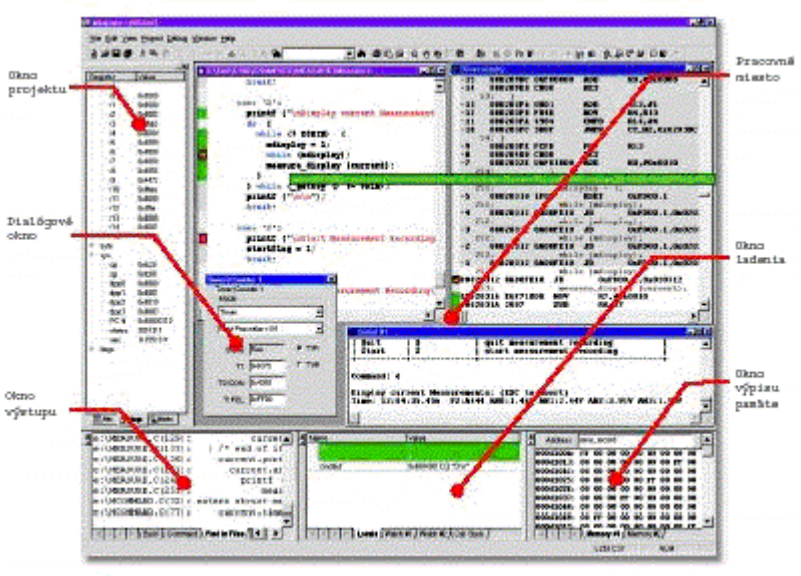
Optimalizácia prebieha pomocou:

- eliminácia mŕtveho kódu,
- presúvanie časti programov k minimalizácií skokov,
- nastavenie premenných do registrových bánk,
- parametrické prechádzania registrov,
- eliminácia podprogramov konajúcich rovnakú funkciu,
- spojovaním inštrukčných blokov k minimalizácií skokov,
- vytváranie podprogramov z častí programu tvoriacich rovnakú funkciu.

Popis ovládania prostredia

µVision2 pracuje v dvoch režimoch:

- **build mode** − pre tvorbu zdrojového kódu a preklad,
- **debug mode** − obsahuje výkonný debugger pre testovanie aplikácií.



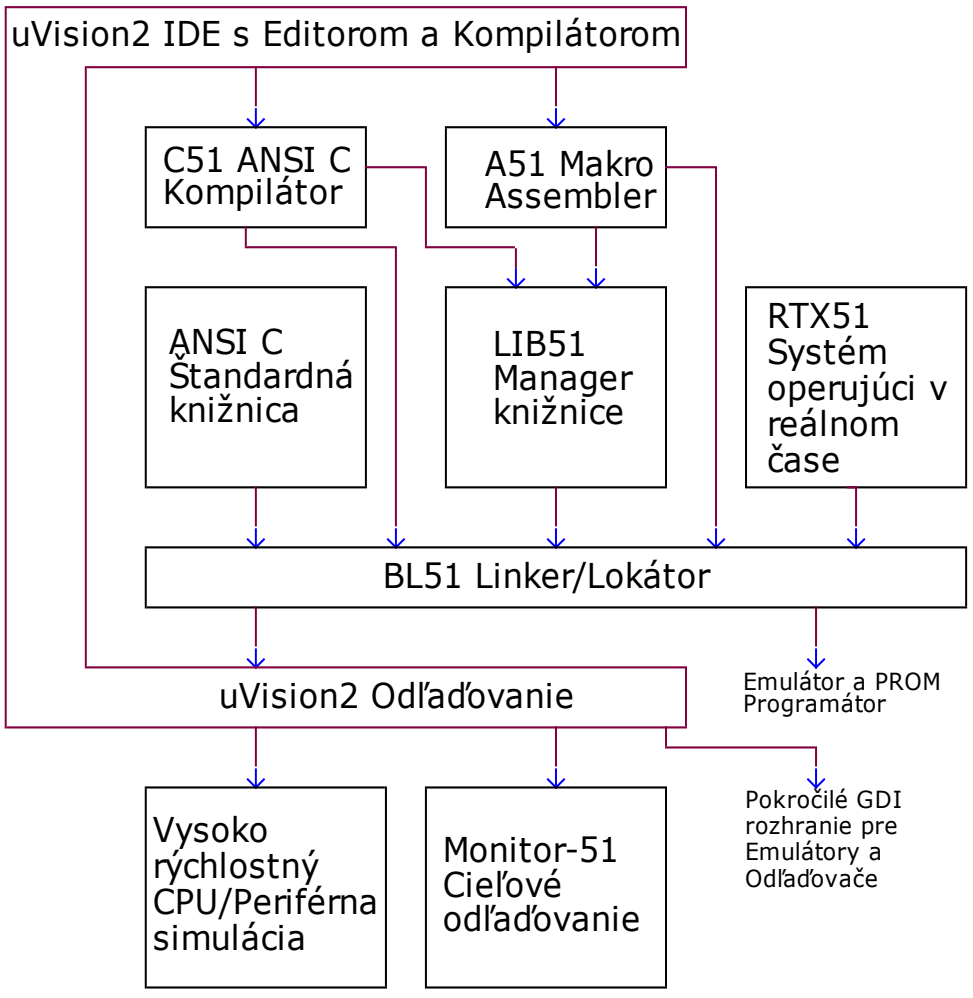
Obr. 1 Pohľad na IDE µVision2

Po spustení sa neobjavia všetky okná ako na hore uvedenom obrázku, ale iba okno Project Window a okno Output Window. Project Window slúži pre správu projektu. V záložke Files sú zobrazené všetky súbory používané k prekladu. Tieto

súbory sú zoraďované do skupín (vyjadrených adresárom), kde každá skupina môže mať rôzne nastavené atribúty pre preklad. Ako výhodné sa javí zoraďovanie programov do funkčných skupín, napríklad pre identifikáciu inžiniera v softvérovom týme. V záložke Books sú súbory nápovede, kde je uložené všetko, čo si programátor práve nepamätá (napr. Datasheet).

Záver

µVision2 IDE i keď je schopný sám vytvoriť veľmi kvalitný kód, sa dodáva ako súčasť softvérového balíka. Sú štyri verzie týchto balíkov a je možné ich objednať cez www.hwgroup.cz. Popísanie ostatných programov v balíku nie je cieľom tohto článku.



Obr. 2 Štruktúra µVision2

Pre začiatok na naučenie je možné z web servera firmy Keil stiahnuť výukovú verziu. Táto verzia má určité obmedzenia (nepodporuje nejaké zložitejšie inštrukcie, ladiť je možné program iba menší než 2 kB atď.) Jedná sa bezpochyby

o špičkový, profesionálny softvér, ktorý samozrejme stojí (nemalé) peniaze. Je však nesporné, že pri vývoji ušetrí mnoho času a tak sa tieto peniaze skoro vrátia. Keď k tomu uvážime, že X51 je najrozšírenejšia a so všetkými kompatibilná, dostáva sa nám do rúk veľmi výkonný nástroj.

2. CodeVisionAVR − ANSI C kompilátor

Firma ATMEL vyrába osembitové RISC mikroradiče AVR, ich jadro je optimalizované pre použitie prekladačov C. Firma ATMEL sama žiadny prekladač C neposkytuje, zato poskytuje zdarma AVR Studio vrátane assembleru. To je však navrhnuté tak, že môže spolupracovať s prekladačmi C niekoľkých výrobcov.

CodeVisionAVR je veľmi výkonný kompilátor jazyka C, integrované vývojové prostredie a programátor In-System pre rodinu mikrokontrolérov Atmel AVR s internou pamäťou RAM. Kompilátory CodeVision je možné objednať na http://shop.hw.cz.

CodeVisionAVR je taktiež známy ako CVAVR alebo jednoducho CodeVision. Tento kompilátor pochádza od Pavla Haiduca z HP Infotech S.R.L. Jedná sa o plnohodnotný vývojový systém pre mikrokontroléry Atmel AVR, dostupný v dvoch verziách − plná (Standard), ktorá generuje kód pre všetky klasické AVR a ATMega série, a odľahčená (Light) verzia, ktorá vytvára kód iba pre klasickú AVR rodinu (čísla súčiastok začínajúce AT90S...). K dispozícií bude tiež verzia pre súčiastky bez statickej pamäti SRAM (TinyAVR a AT90S1200) nazývaná CodeVisionAVR Tiny. Táto verzia je k stiahnutiu bezplatne z webu http://www.hpinfotech.ro a je obmedzená veľkosťou programu, ktorý je možné skompilovať. Limit veľkosti programu však stále ešte dovoľuje veľmi slušnú prácu so systémom. Pre vyskúšanie programu je k dispozícií skúšobná verzia s obmedzenou veľkosťou výsledného kódu.

Veľká obľuba vývojových prostriedkov CodeVisionAVR je bezpochyby daná taktiež ich nízkou cenou, resp. skvelým pomerom cena/výkon.

Pre zaujímavosť uvádzam ceny produktov CodeVisionAVR:

- CodeVisionAVR ***Standard*** − cca **6150 SK** bez DPH,
- CodeVisionAVR ***Light*** − cca **3690 SK** bez DPH,
- CodeVisionAVR ***Tiny*** − bezplatná, zatiaľ nezverejnená verzia.

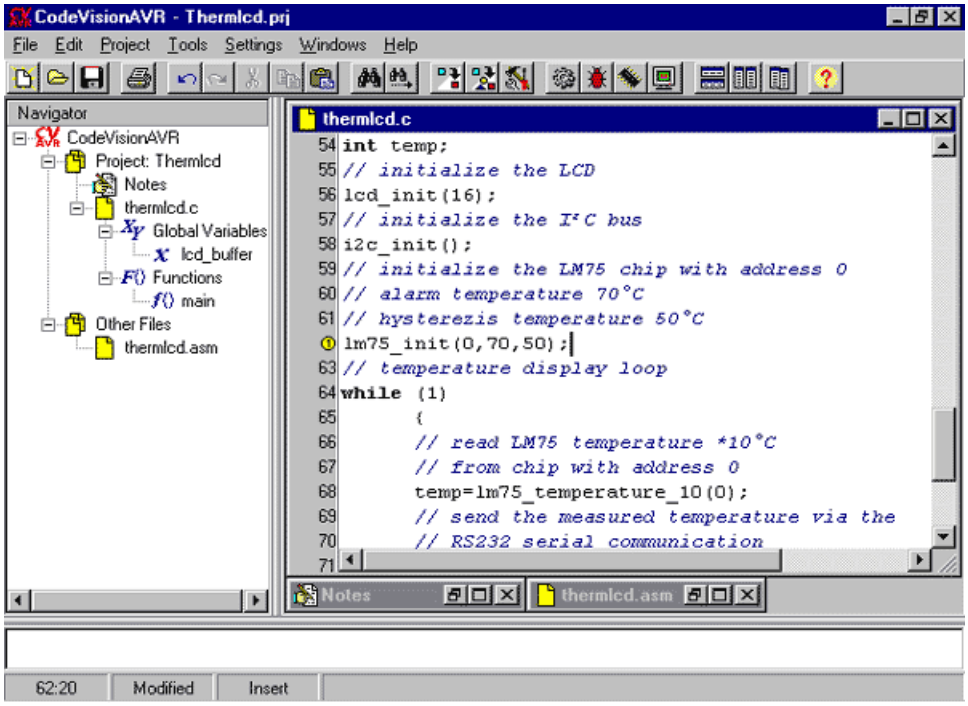
Bezplatná skúšobná verzia CodeVisionAVR je rovnaká s komerčnou verziou, okrem toho, že veľkosť kompilovaného kódu je obmedzená a knižnice pre PCF8563, PCF8583, DS1302 a DS1307 RTCS sú odstránené.

Základné vlastnosti

- 32-bitová aplikácia, spustiteľná pod Windows 95/98/NT4.0/2000/XP,
- integrované vývojové prostredie a kompilátor jazyka C s ľahkým použitím,
- editor s automatickým odrážkovaním a zvýraznením syntaxe,
- podpora dátových typov:
  - bit,
  - char,
  - int,
  - short,
  - long,
  - float,
- podpora špecifických rozšírení mikrokontrolérov AVR:
  - prístup k pamäťovým poliam EEPROM a FLASH,
  - prístup k registrom na bitovej úrovni,
  - podpora prerušení,
- rozsiahle možnosti optimalizácie výsledného kódu kompilátora vrátane:
  - odstránenie prebytočného kódu,
  - optimalizácia pre pamäťové modely Tiny (8-bitový ukazateľ pre obvody s pamäťami RAM do 256 bajtov) a Small (16-bitový ukazateľ pre obvody s viac než 256 bajtov RAM),
  - voľby optimalizácie výsledného kódu pre rýchlosť alebo veľkosť programu,
- možnosť vkladania assembleru do zdrojového kódu C,
- ladenie aplikácie na úrovni zdrojového kódu C s výstupom do formátu COFF umožňujúci využitie I/O terminálu z debuggeru Atmel AVR Studia,
- plne zlúčiteľný s emulátormi In-Circuit ATMEL ICE200, JTAG-ICE a ďalšími,
- vstavaný sériový komunikačný terminál RS232, RS422, RS485 pre ladenie aplikácií,
- vstavaný programátor In-System pre AVR s automatickým programovaním po úspešnej kompilácií, kompatibilnou s:
  - ATMEL STK500/STK501/STK502/AVRISP,
  - Kanda Systems STK200+ a STK300,
  - Vogel Elektronic VTEC-ISP,
  - Dontronics DT006,
  - Tietomyrsky EXB2313,
  - 4Ahead AVR Board 1,
  - Futurlec JR-AVR AT90S2313 a AT90S8535.

Grafické vývojové prostredie CodeVisionAVR je veľmi prívetivé a ľahko sa používa. Je projektovo založené a zahrňuje automatický generátor kódu nazvaný

CodeWizard AVR, ktorý generuje všetku potrebnú inicializáciu kódu pre integrované periférie AVR, rovnako ako niektoré vonkajšie periférie, ktoré sú obsiahnuté v dodávaných knižniciach.



Obr. 3 Pohľad na IDE CodeVisionAVR

Funkcie vstavaného automatického generátora kódu CodeWizardAVR:

- nastavenie prístupu k vonkajšej pamäti,
- inicializácia vstupne/výstupných portov (I/O),
- inicializácia externých prerušení,
- inicializácia čítačov/časovačov,
- inicializácia WatchDog Timer-a,
- inicializácia a nastavenie UART s parametrami 7N2, 7E1, 7O1, 8N1, 8N2, 8E1 a 8O1,
- inicializácia analógového komparátora,
- inicializácia A/D prevodníka,
- inicializácia SPI rozhrania,
- inicializácia I^2C Bus, LM75, DS1621, PCF8563, PCF8583, DS1302 a DS1307,
- inicializácia 1-Wire Bus a DS1820/DS1822,
- inicializácia LCD.

Dodávané knižnice

Dodávané knižnice ponúkajú podporu pre veľký počet obvykle používaných vnútorných i vonkajších zariadení, ako sú LCD displeje, hodiny reálneho času (RTC), teplotné snímače, UART, SPI, atď.

- znakové LCD moduly až 4x40 znakov,
- Philips I^2C Bus,
- teplotný senzor National Semiconductor LM75,
- teplomery Dallas DS1621,
- RTC Philips PCF8563 a PCF8583,
- RTC Dallas DS1302 a DS1307,
- 1-Wire protokol Dallas,
- 1-Wire teplotné čidlo Dallas DS1820/DS1822,
- 1-Wire EEPROM Dallas DS2430/DS2433,
- SPI,
- power management,
- konverzia BCD a Gray kódu.

Obvody podporované vstavaným programátorom

Kompilovaný kód môže byť programovaný do mikrokontroléra s pomocou vstavaného ISP (In System Programming), ktorý je možné použiť po úspešnom zostavení kódu. K dispozícií je taktiež terminálový program dodávaný ako časť CodeVision. Terminál môže posielať a prijímať súbory a taktiež má schopnosť zobraziť príjem dát v kódu HEX alebo ASCII a odoslanie jednotlivých hexadecimálnych hodnôt.

CodeVision môže produkovať objektový kód vo formáte COFF, vďaka ktorému je možné pri vývoji aplikácie použiť plnohodnotný debugger z AVR Štúdia od Intelu. Výstupom však môže byť i formát OBJ alebo Intel HEX pre použitie s iným programátorom FLASH a EEPROM.

Nie všetky obvody podporované kompilátorom je možné programovať priamo z ImageCraft. V súčasnej dobe programátor podporuje iba nasledujúce obvody:

- AT90S1200,
- ATtiny12/15L/22/26,
- AT90S2313,
- AT90S2323/2343,
- AT90S2333/4433,
- AT90S4414/8515,
- AT90S4434/8535,
- ATmega603/103,
- ATmega64/128,
- ATmega161,
- ATmega162,
- ATmega163,
- ATmega169,
- ATmega32,
- ATmega323,
- ATmega8/16,
- ATmega8515, ATmega8535,
- AT86RF401.

Kompilátorom podporované obvody:

- ATtiny13,
- ATtiny22,
- ATtiny26,
- AT90S2313,
- AT90S2323/2343,
- AT90S2333/4433,
- AT90S4414/8515,
- AT90S4434/8535,
- AT90S8534,
- ATmega603/103,
- ATmega64/128,
- ATmega161,
- ATmega162,
- ATmega163,
- ATmega169,
- ATmega32,
- ATmega323,
- ATmega8/16,
- ATmega8515,
- ATmega8535,
- FPSLIC AT94K05/10/20/40,
- AT43USB355,
- AT76C711,
- AT86RF401.

Záver

CodeVisionAVR je ideálnym vývojovým prostriedkom najmä pre programátorov, ktorí majú radi vysoký komfort práce a nechcú pritom strácať čas prílišným štúdiom a nastavovaním procesorov. To pochopiteľne platí iba v prípade, že príslušný obvod je obsiahnutý v knižnici kompilátora. CodeVisionAVR si môžeme objednať napr. prostredníctvom http://shop.hw.cz.

Light verzia má rovnaké vlastnosti ako verzia Standard, iba nepodporuje obvody ATmega, AT94K FPSLIC a AT43USB355 a obsahuje len 6 mesiacov technickej podpory.

Ostatné vývojové prostriedky na programovanie mikrokontrolérov ATMEL AVR

BASCOM – je vývojové prostredie vrátane prekladača z jazyka, ktorý sa podobá známemu Visual Basicu 6.0. Je produktom firmy MCS Electronics. Obmedzenie je na maximálne 2 kB výsledného kódu (HEX). Výhodou sú špeciálne príkazy podporujúce prácu s LCD displejmi, komunikáciou I^2C, 1-WIRE atď.

GNU_C – kompilátor C, ktorý je možné nainštalovať ako súčasť AVR Studia. Na tento prekladač nie je žiadne časové obmedzenie alebo obmedzenie veľkosti kódu. Je k dispozícii celkom zadarmo. Pre jeho použitie je iba nutné dodržať licenciu GNU.

IAR – obsahuje časovo obmedzené vývojové prostredie firmy IAR. Jedná sa predovšetkým o assembler a prekladač z jazyka C/C++. Konkrétne sa jedná o IAR Embedded Workbench Evaluation version for Atmel AVR v2.27B a IAR Embedded Workbench Assembler Edition for Atmel AVR v1.50B. Na viac je tu umiestnený i produkt IAR MakeApp for Atmel AVR v3.01.

JAVA – klasická Java, ku ktorej sú pridané knižnice JEPES dánskej firmy Mjolner Informatics. Demo verzia umožňuje programovať iba AT90S8515.

PASCAL – ideálny prostriedok na programovanie, jedná sa o školnú verziu produktu (demo) nemeckej firmy E-LAB Computers. Obmedzenie je na maximálne 4 kB výsledného kódu (HEX), čo pre väčšinu aplikácií stačí. V assembleri to predstavuje cca 6000 riadkov kódu.

Atmel – obsahuje predovšetkým vývojové prostredie AVR Studio v4.07, WAVRASM v1.30, ktorý rovnako umožňuje kompletný vývoj programov pre ATMEL AVR v assembleri.

PRÍLOHA 7 − PREHĽAD A OPIS ELEKTRONICKÝCH SÚČIASTOK

Tab 1. Zoznam elektronických súčiastok

Názov:	Mikrokontrolér ATMEL AVR
Typové označenie:	AT90S8535
Označenie v schéme:	IC1
Popis činnosti, funkcia:	Jednočipový programovateľný mikropočítač osadený v puzdre DIP 40. Podrobnejší opis viď. *Príloha 2 a 3*. Riadi celú činnosť vývojovej dosky.
Názov:	Kryštál
Typové označenie:	8 MHz
Označenie v schéme:	QU1
Popis činnosti, funkcia:	Je to oscilátor generujúci frekvenciu 8 MHz potrebnú k chodu mikrokontroléra. Je umiestnený v puzdre HCJ-30.
Názov:	Keramický kondenzátor
Typové označenie:	33 pF
Označenie v schéme:	C1, C2
Popis činnosti, funkcia:	Spolu s QU1 tvoria rezonančný obvod.
Názov:	Odpor
Typové označenie:	1 kΩ
Označenie v schéme:	R1
Popis činnosti, funkcia:	Externá súčiastka na realizáciu RESET-u mikrokontroléra.
Názov:	Pole 8xLED diód
Typové označenie:	8xLED (4x červená, 4x zelená)
Označenie v schéme:	DPACK1
Popis činnosti, funkcia:	Výstupná signalizácia (vizuálna).

Názov:	Odporová sieť
Typové označenie:	8x220 Ω
Označenie v schéme:	RPACK1
Popis činnosti, funkcia:	Predradné rezistory na obmedzenie prúdu LED diódami.
Názov:	Prepínač
Typové označenie:	
Označenie v schéme:	SW DIP-4
Popis činnosti, funkcia:	Prepínanie funkčných častí na jeden port mikrokontroléra.
Názov:	Konektor
Typové označenie:	Cannon 25 F
Označenie v schéme:	CON2
Popis činnosti, funkcia:	Výstup ISP programátora. Pripája sa káblom na LPT port PC.
Názov:	Integrovaný obvod
Typové označenie:	MH 74244
Označenie v schéme:	IC5
Popis činnosti, funkcia:	
Názov:	Odpor
Typové označenie:	100 kΩ
Označenie v schéme:	R20
Popis činnosti, funkcia:	Externá súčiastka pre obvod MH 74244.
Názov:	Keramický kondenzátor
Typové označenie:	100 nF
Označenie v schéme:	C7
Popis činnosti, funkcia:	Filtrácia napájacieho napätia.

Názov:	LED dióda
Typové označenie:	3mm, červená
Označenie v schéme:	D2
Popis činnosti, funkcia:	Indikácia prítomnosti napájacieho napätia.
Názov:	Odpor
Typové označenie:	100 kΩ
Označenie v schéme:	R21
Popis činnosti, funkcia:	Obmedzenie I pre LED diódu D1.
Názov:	Dióda
Typové označenie:	1N4148
Označenie v schéme:	D1
Popis činnosti, funkcia:	Ochrana proti prepólovaniu.
Názov:	Odpor
Typové označenie:	220 Ω
Označenie v schéme:	R2-R8
Popis činnosti, funkcia:	Obmedzenie I pre 7 segmentový displej DISP1.
Názov:	Displej
Typové označenie:	28 TFK 549
Označenie v schéme:	DISP1
Popis činnosti, funkcia:	7 segmentový displej. Výstupná vizualizácia.
Názov:	Tranzistor
Typové označenie:	BC 640
Označenie v schéme:	T1
Popis činnosti, funkcia:	Budenie displeja DISP1 pri dynamickom zobrazovaní.

Názov:	Odpor
Typové označenie:	1 kΩ
Označenie v schéme:	R9
Popis činnosti, funkcia:	Nastavenie budiaceho I pre tranzistor T1.
Názov:	Membránová klávesnica 4x3
Typové označenie:	
Označenie v schéme:	KP1
Popis činnosti, funkcia:	Zadávanie vstupov od užívateľa.
Názov:	Odpor
Typové označenie:	10 kΩ
Označenie v schéme:	R10-R17
Popis činnosti, funkcia:	Obmedzenie I pre tlačidlá TL1-TL8.
Názov:	Tlačidlá
Typové označenie:	Mikrotlačidlá
Označenie v schéme:	TL1-TL8
Popis činnosti, funkcia:	Zadávanie vstupov od užívateľa.
Názov:	LCD
Typové označenie:	HD44780
Označenie v schéme:	LCD1
Popis činnosti, funkcia:	2x16 znakov. Výstupná vizualizácia.
Názov:	Nastaviteľný odpor − Trimer
Typové označenie:	2,5 kΩ
Označenie v schéme:	R18
Popis činnosti, funkcia:	Nastavenie kontrastu pre LCD.

Názov:	Pamäť EEPROM
Typové označenie:	AT24C02
Označenie v schéme:	IC4
Popis činnosti, funkcia:	Externá sériová pamäť EEPROM.
Názov:	A/D prevodník
Typové označenie:	TLC 547
Označenie v schéme:	IC3
Popis činnosti, funkcia:	8 bit A/D prevodník, pracujúci na zbernici MicoWire.
Názov:	Nastaviteľný odpor − Trimer
Typové označenie:	5 kΩ
Označenie v schéme:	R19
Popis činnosti, funkcia:	Nastavenie referenčného napätia pre TLC 547.
Názov:	Prevodník úrovní RS232/TTL
Typové označenie:	MAX 232
Označenie v schéme:	IC2
Popis činnosti, funkcia:	Prevodník úrovní RS232/TTL.
Názov:	Elektrolitický kondenzátor
Typové označenie:	1 µF
Označenie v schéme:	C3-C6
Popis činnosti, funkcia:	Nábojové pumpy.
Názov:	Konektor
Typové označenie:	Cannon 9 F
Označenie v schéme:	CON1
Popis činnosti, funkcia:	Sériová komunikácie s PC.

Názov:	Teplotný snímač
Typové označenie:	DS18B20
Označenie v schéme:	IC6
Popis činnosti, funkcia:	Teplotný snímač, zbernica 1-Wire, citlivosť: +/- 0,5 °C, rozsah teplôt: - 10 do + 85 °C.
Názov:	Odpor
Typové označenie:	4,7 kΩ
Označenie v schéme:	R22
Popis činnosti, funkcia:	

PRÍLOHA 8 − INTELIGENTNÉ DISPLEJE LCD

Úvod

Vďaka rozšíreniu mikroprocesorov v posledných niekoľkých rokoch sa textové a grafické LCD displeje s inteligentnými radičmi stávajú čím viac tým obľúbenejšími. Ich cena sa síce pohybuje rádovo okolo 400 SK, ale za tieto peniaze získava zákazník profesionálne vypadajúci displej s 32 znakmi, ktorý je možné veľmi dobre ovládať práve z mikroprocesora.

Vďaka týmto vlastnostiam sú v amatérskych konštrukciách klasické displeje skladané z niekoľko segmentov stále málo časté. Ich cena je totiž síce nižšia, ale iba pri malom počte číslic. K ich ovládaniu na viac potrebujeme pre každú číslicu zvlášť budič a ďalšie obvody. Pokiaľ použijeme multiplexné zapojenie pre ovládanie klasického displeja, používame s najväčšou pravdepodobnosťou k riadeniu multiplexu opäť mikrokontrolér.

Rozdelenie LCD

LCD displeje môžeme rozdeliť do niekoľkých skupín. Toto delenie použijeme pri výbere vhodného LCD displeja pre našu aplikáciu.

Podľa najmenšej zobrazovacej jednotky:

- **znakové** (najmenšia zobrazovacia jednotka je znak):
  - jednoriadkové (1x8, 1x10, 1x12, 1x16, 1x20, 1x24, 1x40, ...),
  - viacriadkové (2x8, 2x10, 2x16, 2x20, 2x24, 2x40, 4x16, 4x20, 4x24, 4x40, ...),
- **grafické** (najmenšia zobrazovacia jednotka je bod):
  - kombinácia čísel 64, 80, 97, 98, 120, 122, 128, 160, 192, 240, 256, 320, 640,... ,
- **podľa riadiacej matice:**
  - **aktívne** riadiace matice,
  - **pasívne** riadiace matice,
- **podľa podsvietenia displeja:**
  - nepodsvietené,
  - podsvietené,
- **podľa druhu podsvietenia:**
  - LED,
  - EL,
  - CCFL,
- **podľa farby podsvietenia:**
  - biele, žltozelené, modrozelené, jantárové, červené, oranžové, zelené,

- **podľa druhu zobrazenia displeja:**
  - TN (Twisted Nematic),
  - STN (Super Twisted Nematic),
  - FSTN (Film-compensated Super Twisted Nematic),
  - negatívne,
  - pozitívne,
- **podľa typu zdroja svetla:**
  - transparentné (displeje prenosných PC),
  - reflexné (hodinky, kalkulačky),
  - projekčné (projektory s tekutými kryštálmi),
- **podľa druhu konektoru na displeji:**
  - tlačený spoj s otvormi v doske,
  - pin konektor priamy,
  - pin konektor 90 °.

Charakteristika displejov LCD

LC displeje sa vyskytujú vo veľkom množstve aplikácií. Obecne, ich výhodou je malý odber zobrazovacej matice (rádovo desiatky μA), malé rozmery a nízka hmotnosť v porovnaní s klasickou elektrónovou obrazovkou, lepšia geometria a ostrosť zobrazenia, dlhšia životnosť, stálosť obrazu (LC displeje majú oveľa vyššiu frekvenciu obnovenia informácie a ich fyzikálny princíp umožňuje oveľa dlhší "dosvit", takže odpadá klasické blikanie, nešvár to elektrónových obrazoviek a displejov) atď.

Nevýhodou, ktorá je však už u niektorých displejov odstránená (na úkor ceny), je teplotná závislosť, keď kvapalné kryštály pri záporných a vysokých kladných teplotách strácajú svoje fyzikálne vlastnosti a displej prestáva dočasne pracovať.

Porovnanie LED a LCD displejov

Znakové LCD displeje nie sú však tak veľké a majú nižší kontrast, ako LED displeje. Pre určité druhy aplikácií ich preto nie je možné nepoužiť. Existuje však taktiež klasický LCD displej s väčšou výškou znaku. Tieto väčšie displeje síce nie sú v SR a ČR príliš rozšírené, ale pozornosť vývojárov by si iste zaslúžili taktiež. LCD displej má oproti LED číslovkám veľkú výhodu, lebo ho je možné použiť taktiež pre textový výstup. To dáva vaším aplikáciám nový rozmer v intuitívnosti ovládania.

LCD displej pripojený k mikroprocesoru je taktiež výborná pomôcka pri odlaďovaní aplikácií. Rutina ovládajúca LCD displej je totiž celkom jednoduchá a väčšinou funguje na prvé spustenie. Pokiaľ nemáme k dispozícii profesionálne vývojové prostriedky, je možné s jedným tlačidlom a LCD displejom odlaďovať

i celkom zložité programy. Na LCD je možné vypisovať stavy premenných, adresu pamäti, kde sa nachádzame, obsahy registrov atď.

Postrehy práce s LCD displejmi z praxe

Pokiaľ používame inteligentné LCD, narazili sme už pravdepodobne na niekoľko detailov, ktoré je nutné veľmi dobre poznať a mať ich naštudované. Niektoré z týchto vecí tu uvediem, ale v tomto článku im nie je možné venovať detailný priestor.

Znakové sady

LCD displej obsahuje 256 pozícií, v ktorých sú uložené fonty. Časť týchto pozícii je však nedostupná, pretože určitú kombináciu adries sa predávajú displeju konfiguračné a systémové údaje (ZAP/VYP kurzoru, 4/8-bitová komunikácia atď.) Na začiatku je navyše 8 pozícii pre stiahnuteľné fonty. V praxi zostáva 2x96 znakov. Prvá polovica z nich je v mape fontov posunutá tak, aby znaky odpovedali ASCII konvencii = “A” na pozícii 41H alebo 65. Horná polovica znakovej sady je však voliteľná.

Väčšinou sa používa štandardná anglická sada ako prvá (odpovedá ASCII), ale v hornej pozícii môže byť obsiahnutých niekoľko sad, podľa toho existujú displeje:

- anglická/japonská (KS0066F00),
- anglická/japonská (SED1278DOA),
- anglická/európska (KS0066F05),
- anglická/európska (SED1278DOB),
- anglická/ruská,
- a ďalšie.

Pokiaľ sme pripravili aplikáciu na jeden displej a teraz chceme použiť iný, je potrebné prejsť si podrobne mapu znakov i keď je ako spodná polovica uvedená Anglická znaková sada a horných 96 bajtov nevyužívame. Pretože pomerne často sa líši napríklad posledných i niekoľko posledných znakov zo spodných 96 pozícii! Typickým príkladom sú displeje ELATEC (http://www.elatec.cz/), kde sa u verzie anglická/japonská a anglická/ruská líši taktiež 5 znakov v spodnej polovici na adresách 7B,7C,7D,7E a 7F.

Podsvietenie LCD displejov

LCD displeje je možné kúpiť v podsvietenej i nepodsvietenej verzii. Podsvietenie sa realizuje dvoma spôsobmi:

- svietiace fólie,
- pomocou SMD LED s rozptylovým materiálom**.**

Displeje s fóliou sú nižšie, ale s fóliovým podsvietením je spojený problém so špeciálnym budičom pre rozsvietenie fólie. Podstatným problémom fólií je však klesajúca svietivosť s časom. Tento problém je natoľko kritický, že od použitia takto podsvietených displejov sa všeobecne ustupuje.

Podsvietenie pomocou LED nepotrebuje špeciálne napájanie, technológia LED je lacná a osvedčená. Konštrukčne je nutné bodové svetlo z LED rozniesť do konštantnej plochy, čo zaisťuje rozptylová podložka, ktorá však LED podsvietené displeje citeľne zväčšuje. Podľa toho je možné taktiež ľahko poznať nepodsvietený displej, ktorý je podstatne tenší.

LCD a záporné teploty

LCD technológia má štandardne problémy s nízkymi teplotami, keď je treba väčšie elektrické pole pre natáčanie tekutých kryštálov. Vďaka štandardnej veľkosti napájacieho napätia 5 V je problémom funkcia displeja pri napr. –20 °C. Výrobcovia tento problém vyriešili vyvedením pinu pre riadenie kontrastu na konektor a ďalšie riešenie je na vývojároch. Pokiaľ potrebujeme prevádzkovať displej i pri nižších teplotách, stačí na pin V0 (Supply Voltage for LCD) pripojiť záporné napätie.

Výrobcovia vyrábajú väčšinou niekoľko verzií LCD displejov. Niektoré vyžadujú záporné napätie na V0 a fungujú vo veľkom teplotnom rozsahu, vrátane záporných teplôt, iným stačí V0 = 0 V (pripojenie na svorku GND), ale nefungujú v záporných teplotách. Existujú však i displeje, ktoré sú pri V0 = GND nečitateľné a pin je nutné pripojiť naopak na napájacie napätie.

Poslednou dobou sa presadzuje komerčný štandard, ktorý funguje do –10 °C ale pre bežné teploty vystačí s V 0 = 0 V.
Typickou ukážkou sú práve displeje ELATEC.

LCD a definované fonty

V bežných LCD je 8 pozícií pre užívateľsky definované fonty. Toho je možné využiť pre slovenské znaky, alebo pre rôzne špeciálne znaky, prípadne tzv. semigrafiku, kde pomocou 8 programovateľných znakov zostavíme pseudo obrázok. U podobných obrázkov je treba počítať s medzerami medzi znaky v maske displeja.

LCD displeje založené na radiči HD44780

Bavme sa však teraz o rade displejov s radičom HD44780 a kompatibilnými. Výhodou týchto displejov je rozšírenosť medzi konštruktérmi, relatívne ľahké ovládanie, možnosť definície až osem užívateľských znakov (napr. slovenčina), rôznorodosť vo výberu veľkosti (1x16 až 4x40 znakov) pri zachovaní jednotného ovládania a v neposlednej rade sa ich špecifikácia zavádza ako priemyslový štandard, rovnako zapojenie vývodov (*). To uľahčuje ladenie na HW i SW úrovni a prípadnú výmenu displeje (napr. za väčší, podsvietený a pod.) Za cenu mierneho skomplikovania protokolu je možné s displejom komunikovať buď osem alebo štvorbitovo.

Podstatné maličkosti

Prvých osem znakov v znakovej sade je možné nastaviť, čo umožní na displeji tvoriť rôzne efekty, grafické symboly, slovenčinu alebo animácie. Pre slovenčinu však často osem znakov nepostačuje, čo si vynucuje meniť dynamicky generátor znakov podľa konkrétnej potreby grafických znakov.

Použitá zobrazovacia matica displeja a nastavený režim určujú, či k zobrazeniu užívateľsky definovaných znakov bude použitý raster 5x7 alebo 5x10 pixlov. Veľkosť znakov, ktoré nahrávame je logicky 8x8 pixlov, takže v rozlíšení 5 šírka x 7 výška sa ôsme slovo v generátoru znakov ignoruje a siedme sa používa i ako podtrhnutie znaku. V móde 5x10 je k definícii znaku použité 8 riadkov, deviaty sa nepoužíva a desiaty slúži k zobrazeniu kurzora.

Displej je možné nastaviť (pokiaľ to umožňuje hardvér) do módu jednoriadkového, dvoj alebo štvorriadkového. Bohužiaľ organizácia pamäti rozloženia znakov neumožňuje i cez automatickú inkrementáciu čítača priamy prechod medzi riadky, pretože riadky na seba priamo nenadväzujú. Vďaka tomu je scrollovanie textu po displeji nutné vždy obsluhovať softvérovo.

Význam jednotlivých vývodov displeja

Číslo vývodu	Označenie	V/V	Význam
1	Vss	-	0V (napájanie)
2	Vcc	-	+5V (napájanie)
3	Vee	-	Nastavenie kontrastu
4	RS	Vstup	0 = vstup je inštrukcie 1 = vstup sú dáta
5	R/W	Vstup	0 = zápis dát do LCD 1 = čítanie dát z LCD
6	E	Vstup	Aktivácia displeja
7	DB0	V/V	Dáta, bit 0 (najnižší)
8	DB1	V/V	Dáta, bit 1
9	DB2	V/V	Dáta, bit 2
10	DB3	V/V	Dáta, bit 3
11	DB4	V/V	Dáta, bit 4
12	DB5	V/V	Dáta, bit 5
13	DB6	V/V	Dáta, bit 6
14	DB7	V/V	Dáta, bit 7 (najvyšší)
15 (**)		-	Napájanie podsvietenie, anóda
16 (**)		-	Napájanie podsvietenie, katóda

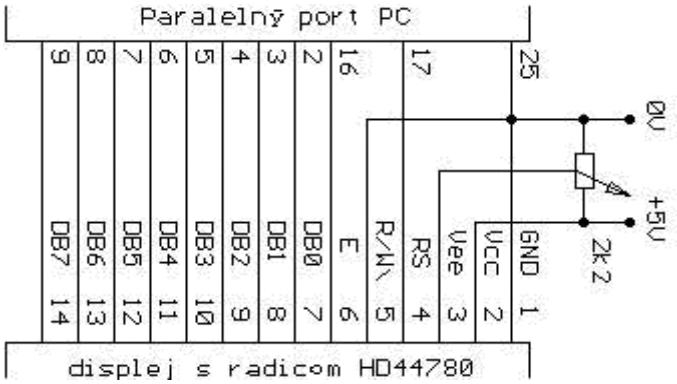
(*) Pred prvým pripojením displeja si skontrolujme zapojenie vývodov v dokumentácii výrobcu ku konkrétnemu displeju!

(**) Vývody nemusia byť osadené, ak nemá displej podsvietenie.

Pripojenie k PC

Tu uvedené pripojenia sú iba jednou z mnohých možností. Slúžia však ako názorný príklad a návod pre konštrukciu. Oba spôsoby boli vyskúšané a odladené.

8-bitový interface (rozhranie):

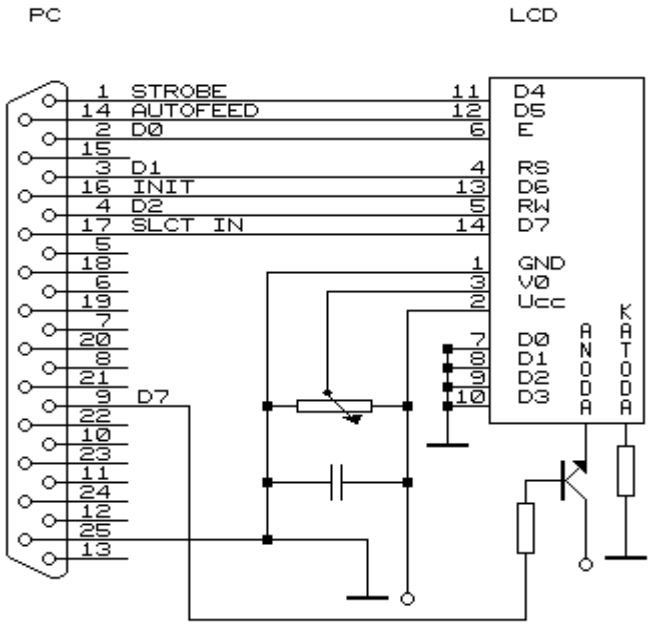


Obr. 1 8-bitový interface (rozhranie) LCD s PC

Priložený softvér obsahuje zazipovanú jednotku pre Borland Pascal verzie 7.0. Jednotka je koncipovaná ako objekt, čo umožňuje jej ľahkú modifikáciu a viacej násobné použitie. Komentovaná by mala byť na úrovni zdrojového textu. Ospravedlňme prosím jej nedokonalosť.

http://www.hw.cz/docs/lcd_iq_displaye/lcd_src.zip, LCD_SRC.ZIP (3231 bajtov)

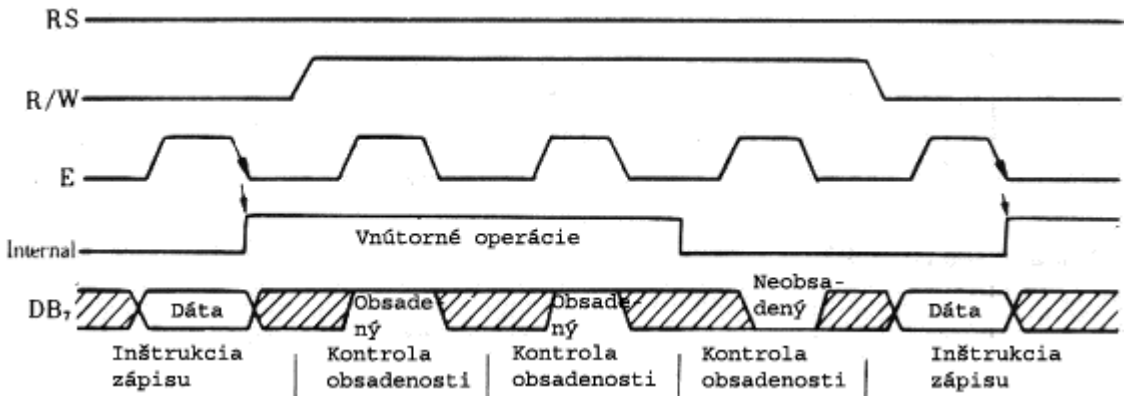
4-bitový interface (rozhranie):



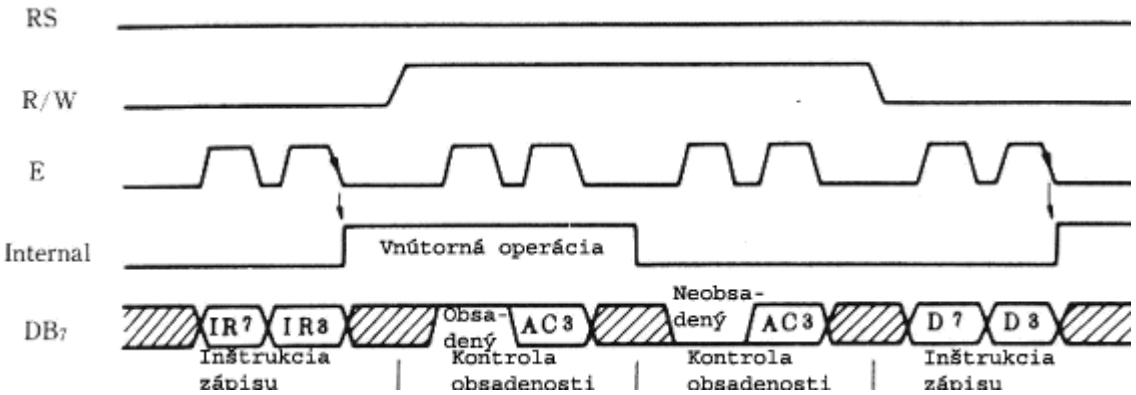
Obr. 2 Štvorbitový interface (rozhranie) LCD s PC

Jan Parkman pre toto zapojenie vyvinul softvér, ktorý obsahuje program pre interaktívne riadenie displeja, editor užívateľsky definovaných znakov a niekoľko utilít. Balík obsahuje i zdrojové kódy hore uvedených programov. Nájdeme ho na: http://www.hw.cz/docs/lcd_iq_displaye/parkman.exe, PARKMAN.EXE (58 062 bajtov)

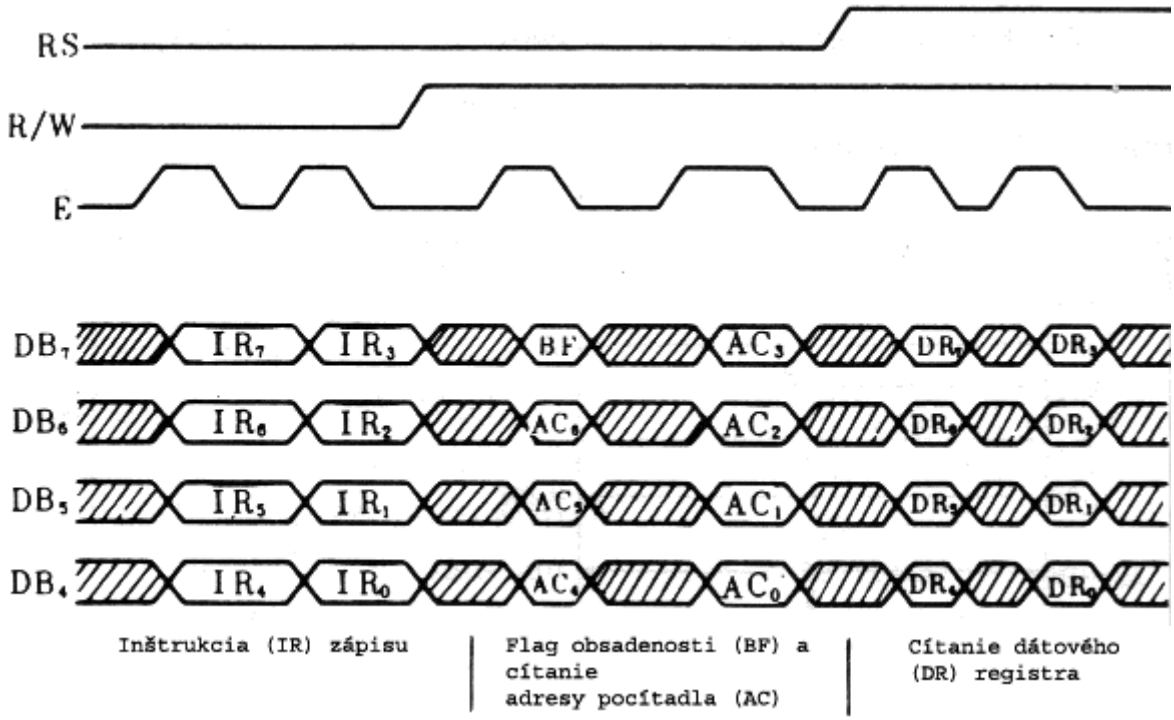
Komunikačné protokoly



Obr. 3 Komunikácia riadiaceho systému s displejom po osembitovej zbernici



Obr. 4 Komunikácia riadiaceho systému s displejom po 4-bitovej zbernici



Obr. 5 Postupnosť posielaných slabík pri štvorbitovej komunikácii

Poznámka: Po zapnutí sa displej nachádza v režime osembitovej komunikácie. Aby bolo možné s ním komunikovať štvorbitovo, je nutné poslať mu najskôr riadiacu sekvenciu (0100 xxxx) "akoby osembitovo" a potom s ním je možné komunikovať štvorbitovo.

Organizácia pamäti radiča

Pamäť displeja pre uloženie pozície znakov na zobrazovacom médiu sa označuje ako DDRAM (Display Data Random Acces Memory). Dole sú uvedené pozície, na ktorých sú uložené znaky pre zobrazenia na riadkoch (nižšia adresa odpovedá znaku viacej vľavo).

Jednoriadkové displeje:

Počet znakov	Pozície v DDRAM
1x8	00h..07h
1x16	00h..0Fh
1x20	00h..13h
1x24	00h..17h
1x32	00h..1Fh
1x40	00h..27h

Poznámka: Niektoré jednoriadkové displeje sa musia inicializovať ako dvojriadkové. Potom i pozícia jednotlivých znakov sa líši od klasického rozloženia. Bližšie informácie nájdete v datasheet-u každého displeja.

Dvojriadkové displeje:

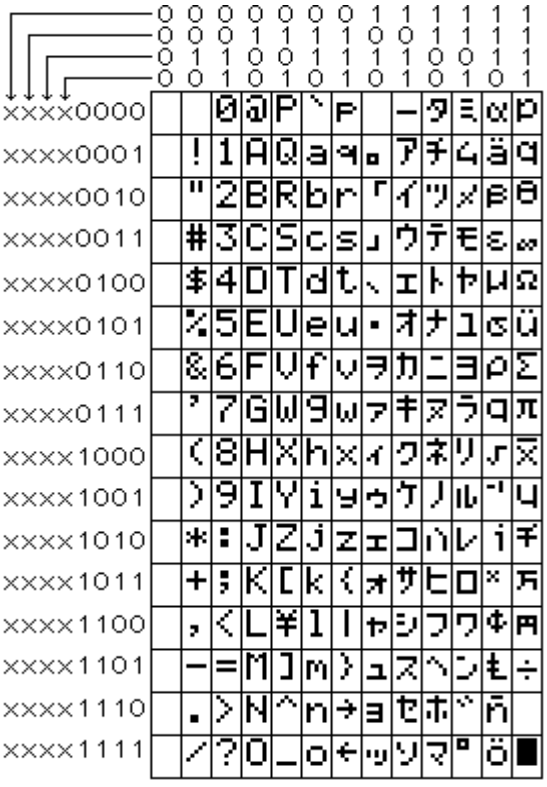
Počet znakov	Pozícia v DDRAM (po riadkoch)
2x16	00h..0Fh 40h..4Fh
2x20	00h..13h 40h..53h
2x24	00h..17h 40h..57h
2x32	00h..1Fh 40h..5Fh
2x40	00h..27h 40h..67h

Štvorriadkové displeje:

Počet znakov	Pozície v DDRAM (po riadkoch)
4x16	00h..0Fh 40h..4Fh 14h..23h 54h..63h
4x20	00h..13h 40h..53h 14h..27h 54h..67h

Znakový generátor

Väčšina displejov používa tu uvedenú znakovú sadu, odlišujú sa mutácie pre najrôznejšie abecedy príliš odlišné od latinky (azbuka..), podrobné informácie potom nájdeme v datasheet-u ku konkrétnemu typu.



Obr. 6 Znaková sada pre LCD s radičom HD44780

Príkazy radiča displeja

Príkaz	Kód										Popis	Dĺžka vykonania príkazu**
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
Zmaže displej	0	0	0	0	0	0	0	0	0	1	Zmaže displej a nastaví kurzor na pozíciu 0.	1,64 mS
Nastaví kurzor na začiatok	0	0	0	0	0	0	0	0	1	*	Nastaví kurzor na pozíciu 0 a vynuluje posun displeja	1,64 mS
Nastaví vstupný režim	0	0	0	0	0	0	0	1	I/D	S	Určí smer pohybu kurzora (I/D) a posun displeja (S). Tieto operácie sa vykonávajú behom čítania/zápisu.	40 µS
Zapne/vypne displej, kurzor a jeho blikanie	0	0	0	0	0	0	1	D	C	B	Zapína/vypína displej (D), kurzor (C) a jeho blikanie (B).	40 µS
Nastaví pohyb kurzora/displeja	0	0	0	0	0	1	S/C	R/L	*	*	Nastaví pohyb kurzora alebo displeja (S/C) a smer pohybu (R/L). Obsah DDRAM zostane bezo zmeny.	40 µS
Nastavenie interface (rozhranie)	0	0	0	0	1	DL	N	F	*	*	Nastaví dĺžku interface (DL), počet riadkov displeja (N) a znakový font (F).	40 µS
Nastaví pozíciu	0	0	0	1	Adresa v CGRAM						Po tomto príkaze	40 µS

v CGRAM						sú dáta zo vstupu zaznamenávané do CGRAM namiesto DDRAM.	
Nastaví pozíciu v DDRAM	0	0	1	Adresa v DDRAM		Po tomto príkaze sú dáta zo vstupu zapisované do a čítané z DDRAM.	40 µS
Číta príznak BUSY a hodnotu adresného čítača	0	1	BF	DDRAM adresa		Číta príznak BUSY (BF) indikujúci, že displej ešte prevádza niektorú operáciu, a pozíciu ukazateľa adresy .	0 µS
Zapíše do DDRAM alebo CGRAM.	1	0	Dáta			Zapíše dáta zo vstupu DDRAM alebo do CGRAM.	40 µS
Číta dáta z DDRAM alebo z CGRAM.	1	1	Dáta			Číta dáta z aktuálnej adresy DDRAM alebo CGRAM.	40 µS

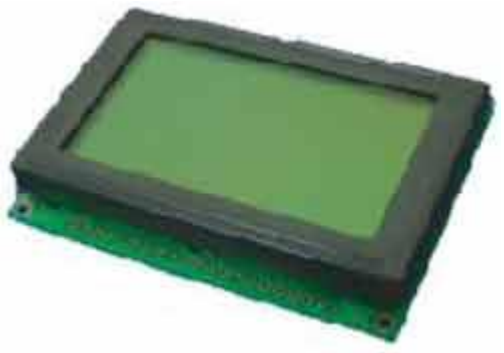
Názov bitu	**Význam nastavenia**	
I/D	0 = Pohyb kurzoru späť	1 = Pohyb kurzora dopredu
S	0 = Posun displeja sa nevykonáva	1 = Displej sa posúva
D	0 = Vypnutie displeja	1 = Zapnutie displeja
C	0 = Vypnutie kurzora	1 = Zapnutie kurzora
B	0 = Vypnutie blikania kurzora	1 = Zapnutie blikania kurzora
S/C	0 = Pohyb kurzora	1 = Pohyb displeja
R/L	0 = Posun doľava	1 = Posun doprava
DL	0 = 4-bit interface (rozhranie)	1 = 8-bit interface (rozhranie)
N	0 = strieda 1/8 alebo 1/11 (jednoriadkový displej)	1 = strieda 1/16 (dvojriadkový displej)
F	0 = 5x7 bodov	1 = 5x10 bodov
BF	0 = Radič prijíma inštrukcie	1 = Radič je zaneprázdnený vykonávaním predchádzajúcej operácie

Grafický LCD displej za cenu znakového

Ako zobrazovacia jednotka v menších prístrojoch s vysokým tlakom na cenu sa často používajú dvojriadkové šestnásťznakové displeje. Grafický displej by v rovnakej aplikácii priniesol viacej flexibility zobrazenia a užívateľského komfortu, ale jeho použitie nebolo doposiaľ možné z dôvodu vysokej ceny. V súčasnej dobe je už situácia iná. Na viac si tento displej môžeme objednať v HW Shope.

Cena malých grafických displejov klesla takmer na hladinu cien znakových displejov pri zachovaní všetkých výhod, ktoré grafické displeje prinášajú. Jedným takýmto príkladom je grafický displej EL16032A. Jedná sa o plne grafický displej s rozlíšením 160x32 bodov, jeho veľkosť zobrazovanej plochy je takmer zhodná s často používaným znakovým displejom s 2x16 znakmi a veľkosťou znaku 9,55 mm, napr. PC 1602 L, EL 1602 S apod.

EL16032A je bežný grafický displej so žltozeleným podsvietením LED v prevedení STN. O komunikácii s okolím sa stará inteligentný radič Sitronix ST7920 obsahujúci čínsku a anglickú znakovú sadu vrátane podpory pre generovanie vlastnej znakovej sady. Súčasťou radiča je pamäť pre zobrazované údaje.

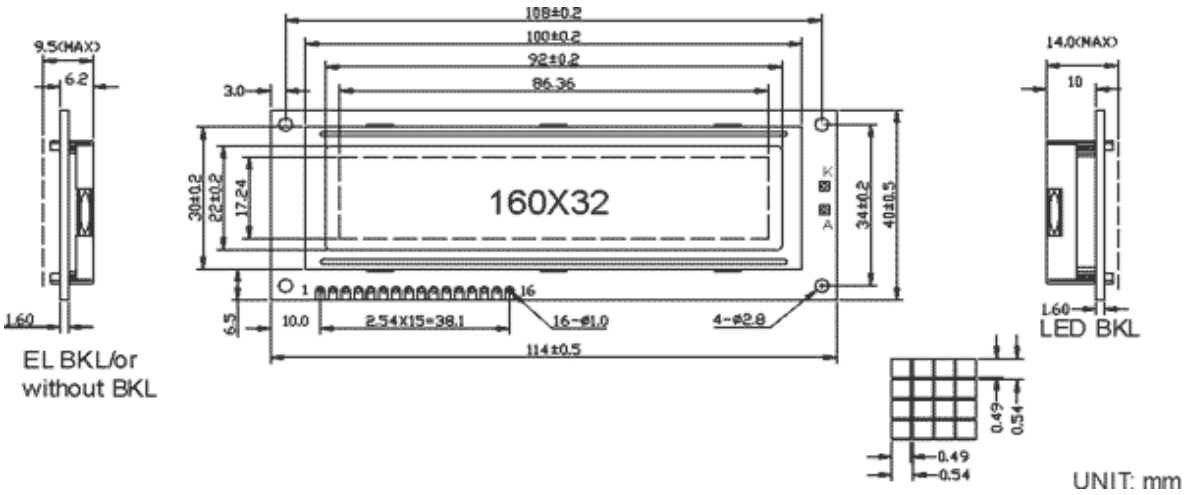


Obr. 7 LCD EL16032A

Displej môže pracovať buď v znakovom, grafickom alebo kombinovanom režime. Je ním možno nahradiť akýkoľvek znakový displej s výhodou zobrazovania grafických symbolov kdekoľvek na ploche. V znakovom režime je možné podľa veľkosti znaku zobraziť 16x4 alebo 16x2 znakov. Sú pridané funkcie pre zjednodušenie komunikácie – zmazanie displeja, návrat na východiskovú pozíciu, displej ON/OFF, kurzor ON/OFF, blikanie znaku, posun znaku, posun zobrazovaného textu na displeji a vertikálne rolovania obrazu.

V grafickom režime je k dispozícii 160x32 bodov. Grafický režim umožňuje podstatne zlepšiť a sprehľadniť obsluhu prístroja, pretože jednotlivé funkcie a údaje môžu byť zobrazené formou prehľadných grafických symbolov, je možné zobrazovať inverzne apod.

V kombinovanom režime môže byť časť displeja prevádzkovaná v riadkovom režime a časť v grafickom podľa potreby. Inteligentný radič generuje pre displej automatický RESET po zapnutí napájania a podporuje 4 alebo 8-bitové pripojenie vrátane pripojenia k dátovej zbernici mikroprocesora. Displej sa vyrába v komerčnom a priemyslovom teplotnom rozsahu.



Obr. 8 Konštrukčné riešenie LCD EL16032A

Displej EL 16032 A je natoľko univerzálny, že ho nie je problém zabudovať bez väčších úprav softvéru alebo mechanického riešenia do vznikajúcich aplikácií a nahradiť ním bežne používané znakové displeje. Aplikácia s grafickým displejom zvyšuje užívateľský komfort a konkurenciu schopnosť výrobku. A to všetko za veľmi priaznivé ceny.

Ďalšie údaje na http://www.elatec.cz/
Ing. Radek Půlpan
Elatec s.r.o.

DOWNLOAD & Odkazy

- domovská stránka Elatec s.r.o. – http://www.elatec.cz/,
- katalógový list EL 16032 A – el16023a_full.pdf,
- objednať EL16032A v HW Shope – 740 SK vrátane DPH,
- ďalšie displeje v HW Shope.

Grafický LCD 128x64 bodov v modrej farbe

Užívatelia sa začínajú odkláňať od klasických žltozelených prevedení a stále častejšie sa stretávame s požiadavkami na rôzne "farebné" prevedenie LCD displejov. Pre podsvietenie sa používajú napr. biele alebo modro svietiace LED alebo CCFL výbojky a kúzli sa s farbami LCD podkladu. Súčasným hitom sú modré displeje.

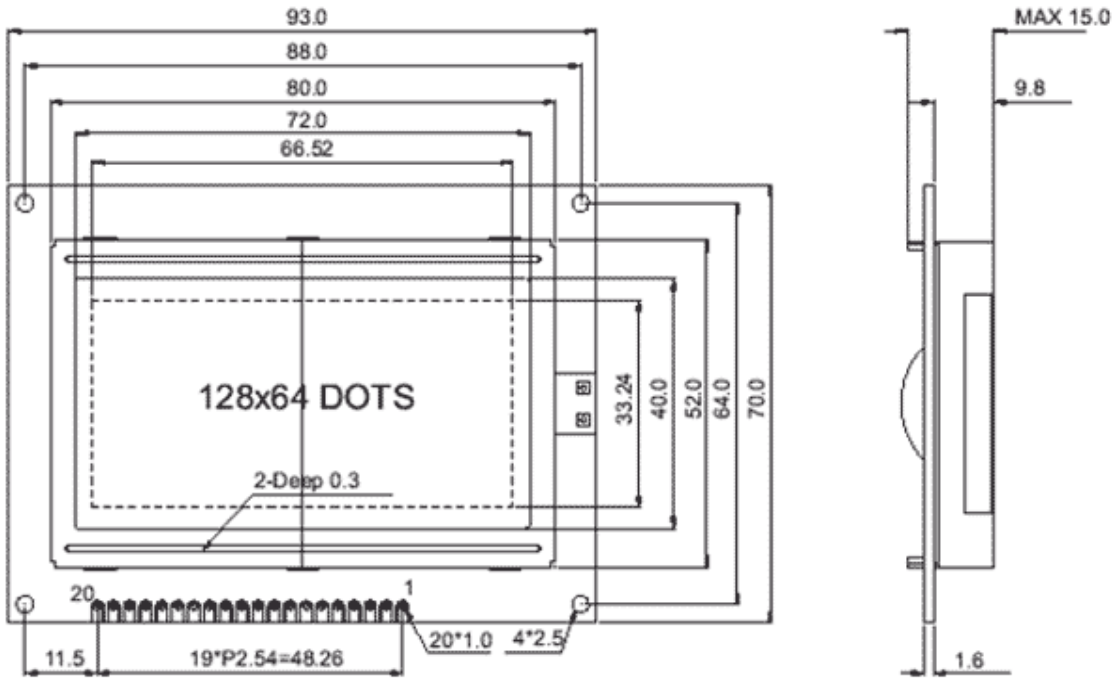
Displej PG 12864 WRM-KNN-IL3 taiwanského výrobcu Powertip patrí k tomu najlepšiemu, čo je možné v súčasnej dobe technologicky u monochromatických

displejov dosiahnuť. Displej je podsvietený bielymi vysoko svietivými LED a je použitá inverzná navy blue technológia. To znamená, že displej má celoplošne modrý podklad a body, ktoré majú byť vidieť, "svieti" bielo. Tým je dosiahnuté príjemného modrého vzhľadu a displej má vysoký kontrast.

Obr. 9 LCD PG 12864

Obecne je grafický displej 128x64 bodov logickým nástupcom riadkových LCD 4x20 znakov v nových konštrukciách, pretože má podobnú veľkosť okienka a riadenie displeja nie je komplikované. Displej zobrazí grafiku 128x64, čo môže byť napr. i text až 8 riadkov po 20-znakoch. Grafický režim však poskytuje obsluhe podstatne vyšší užívateľský komfort.

K riadeniu sú použité dva radiče Samsung KS 107, z nich každý obsluhuje jednu polovicu displeja. Radiče majú implementovanú pamäť RAM pre uloženie bitovej mapy. Po zapísaní obrázku do pamäti radiča sa obrázok generuje na displeji sám bez nutnosti ďalšej obsluhy.



Obr. 10 Konštrukčné riešenie LCD PG 12864

Konštrukčné riešenie LCD PG 12864

Displej je jednoducho pripojiteľný k ľubovoľnému mikroprocesoru. Podrobný popis riadenia displeja typu 128x64 bodov a pripojenie k mikroprocesoru je možné nájsť na http://www.hw.cz/ v čitateľskom servise dodávateľa, firmy Elatec.

Ing. Libor Pavel
pavel@ elatec.cz
Elatec s.r.o.

DOWNLOAD & Odkazy

- domovská stránka Elatec s.r.o. − http://www.elatec.cz/ ,
- domovská stránka výrobcu Powertip − http://www.powertipusa.com/ ,
- katalógový list produktu k stiahnutiu − pg12864wrm.pdf .

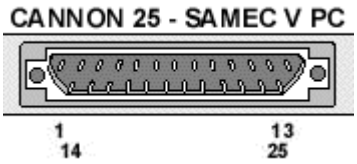
PRÍLOHA 9 – RS232

Úvod

RS232 je rozhranie pre prenos informácií vytvorené pôvodne pre komunikáciu dvoch zariadení do vzdialenosti 20 m. Pre väčšiu odolnosť proti rušeniu je informácia po prepojovacích vodičoch prenášaná väčším napätím, ako je štandardných 5 V. Prenos informácií prebieha asynchrónne, pomocou pevne nastavenej prenosovej rýchlosti a synchronizácia je zostupnou hranou štartovacieho impulzu.

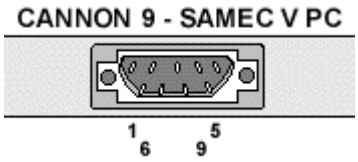
Zapojenie konektorov pre RS232

Cannon 25			
PIN	**NÁZOV**	**SMER**	**POPIS**
1	SHIELD	---	Shield Ground
2	TXD	-->	Transmit Data
3	RXD	<--	Receive Data
4	RTS	-->	Request to Send
5	CTS	<--	Clear to Send
6	DSR	<--	Data Set Ready
7	GND	---	System Ground
8	CD	<--	Carrier Detect
9-19	N/C	-	-
20	DTR	-->	Data Terminal Ready
21	N/C	-	-
22	RI	<--	Ring Indicator
23-25	N/C	-	-



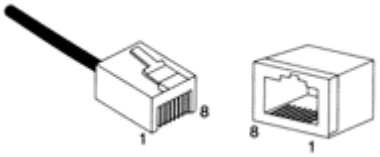
Obr. 1 Cannon 25 – Samec v PC

Cannon 9			
PIN	**NÁZOV**	**SMER**	**POPIS**
1	CD	<--	Carrier Detect
2	RXD	<--	Receive Data
3	TXD	-->	Transmit Data
4	DTR	-->	Data Terminal Ready
5	GND	---	System Ground
6	DSR	<--	Data Set Ready
7	RTS	-->	Request to Send
8	CTS	<--	Clear to Send
9	RI	<--	Ring Indicator



Obr. 2 Cannon 9 – Samec v PC

RJ45			
PIN	**NÁZOV**	**SMER**	**POPIS**
1	RI	<--	Ring Indicator
2	CD	<--	Carrier Detect
3	DTR	-->	Data Terminal Ready
4	GND	---	System Ground
5	RxD	<--	Receive Data
6	TxD	-->	Transmit Data
7	CTS	<--	Clear to Send
8	RTS	-->	Request to Send



Popis signálov

Signál	Popis
DCD – Data Carrier Detect	Detekcia nosnej (niekedy len "CD"). Modem oznamuje terminálu, že na telefónnej linke detekoval nosnú frekvenciu.
RXD – Receive Data	Tok dát z modemu (DCE) do terminálu (DTE).
TXD – Transmit Data	Tok dát z terminálu (DTE) do modemu (DCE).
DTR –Data Terminal Ready	Terminál týmto signálom oznamuje modemu, že je pripravený komunikovať *).
SGND – Signal Ground	Signálová zem.
DSR – Data Set Ready	Modem týmto signálom oznamuje terminálu, že je pripravený komunikovať *).
RTS – Request to Send	Terminál týmto signálom oznamuje modemu, že komunikačná cesta je voľná *).
CTS – Clear to Send	Modem týmto signálom oznamuje terminálu, že komunikačná cesta je voľná *).
RI – Ring Indicator	Indikátor zvonenia. Modem oznamuje terminálu, že na telefónnej linke detekoval signál zvonenia.

* Najskôr plnili riadiace "handshakové" signály funkcie akýchsi "semaforov" pre riedenie polo duplexnej komunikácie s modemami. V plne duplexných komunikačných zariadeniach strácajú riadiace signály čiastočne svoj pôvodný význam a programy ich využívajú skôr spôsobom "zariadenie DTE hlási, že je momentálne pripravené (nepripravené) prijať dáta" **–** k tomu môže programátor využiť ako signál DTR, tak signál RTS – a práve tak je možné pre zariadenie DCE obdobne použiť signál DSR alebo CTS.

Preto niektoré programy umožňujú zvoliť, ktoré handshakové signály sa pre riadenie toku dát použijú **–** v nastaveniach takýchto programov môžeme preto niekedy nájsť viacej variant hardvérového riadenia toku **–** "hardvér (RTS/CTS)", "hardvér (DTR/DSR)" apod. Takto použitý riadiaci signál sa potom podľa nových definícií nazýva správne Ready for receiving.

Podrobný popis signálov na: http://www.hw-server.com/rs232_signals.html .

Základné parametre RS232

Napäťové úrovne

RS 232 používa dve napäťové úrovne. Logickú 1 a 0. Log. 1 je niekedy označovaná ako **marking state** alebo taktiež pokojový stav, log. 0 sa prezýva **space state**.
Log. 1 je indikovaná zápornou úrovňou, zatiaľ čo logická 0 je prenášaná kladnou úrovňou výstupných vodičov.
Povolené napäťové úrovne sú uvedené v tabuľke.
Najbežnejšie sa pre generovanie napätia používa napäťový zdvojovač z 5 V a invertor. Logické úrovne sú potom prenášané napätím +10 V pre log. 0 a –10 V pre log. 1.

Používané káble

Všetky DTE-DCE káble sú priame a vývody sú prepojené 1:1. DTE-DTE a DCE-DCE káble patria medzi krížené.

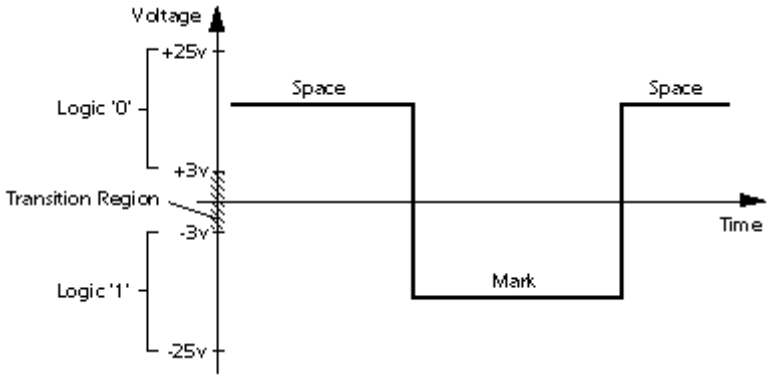
1. **DTE – DCE sa nazýva "Straight Cable" (Priamy)**,
2. **DTE – DTE sa nazýva "Null-Modem",**
3. **DCE – DCE sa nazýva "Tail-Circuit"**.

Problém napájania prepojovaných PC

Pokiaľ prepojujeme dva počítače pomocou RS-232 a každý z nich je pripojený do inej zásuvky, odporúčame zmerať napätie medzi jednotlivými zemami RS-232 pred ich prepojením. Pokiaľ je každý počítač pripojený na inú vetvu i rovnakej fázy, môže byť vplyvom rôznych spotrebičov na každej vetve rozdielové napätie až cca 100 V. To je hodnota, ktorá akýkoľvek RS-232 port spoľahlivo zničí.

Dátové signály		
Úroveň	**Vysielač**	**Prijímač**
Log. L	+5 V do +15 V	+3 V do +25 V
Log. H	-5 V do -15 V	-3 V do -25 V
Nedefinovaný	-3 V do +3 V	

Riadiace signály		
Signál	**Driver (ovládač)**	**Terminátor**
"Off"	-5 V do -15 V	-3 V do -25 V
"On"	5 V do 15 V	3 V do 25 V



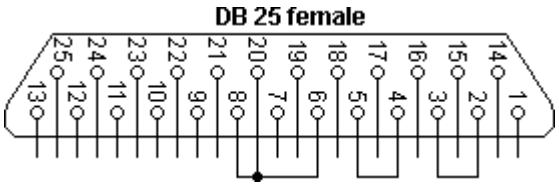
Obr. 3 Napäťové úrovne pri RS232

Zapojenie slučky (LOOPBACK)

Slučka pre testovanie sériového portu

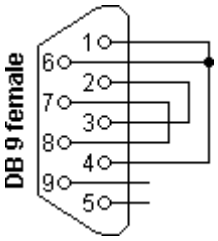
Takto zapojené konektory je možné použiť ku skúšaniu sériového rozhrania na vašom počítači. Sú vzájomne prepojené signály pre prenos dát a riadenia toku. Všetky dáta tak budú ihneď posielané naspäť. Tak je možné ľahko overiť činnosť sériového portu pomocou štandardného terminálového programu.

25-pinový konektor



Obr. 4 DB 25 Samica

9-pinový konektor

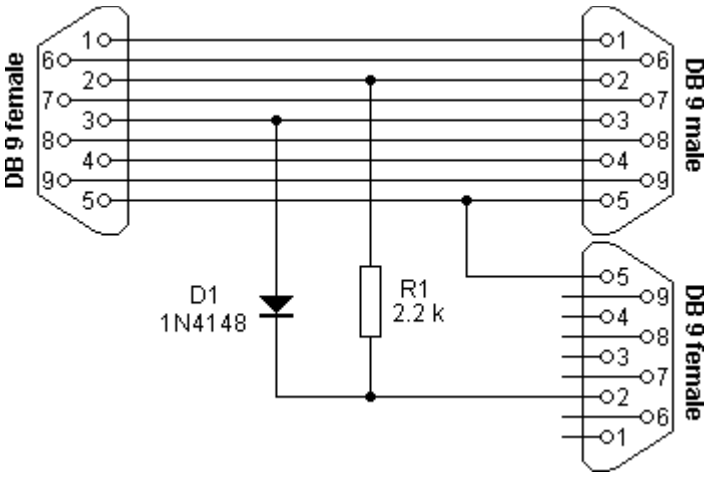


Obr. 5 DB 9 Samica

Monitor RS232

Half Duplex

Monitor RS-232 sériovej komunikácie medzi dvoma zariadeniami s PC. K tejto činnosti je možné využiť nasledujúce zapojenie prepojovacieho káblu. Dva konektory sú spojené priamym káblom. Kontrolný počítač je pripojený k tretiemu konektoru. Tento monitorovací kábel sleduje komunikáciu z oboch zdrojov na jednom RS-232 porte.



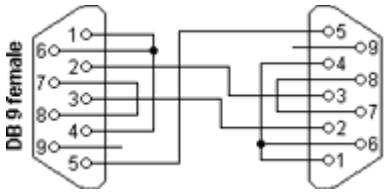
Obr. 6 Monitor RS232

Spy (monitorovací) Softvér

Pre sledovanie prevádzky na sériovej linke je na viac potrebný príslušný softvér.

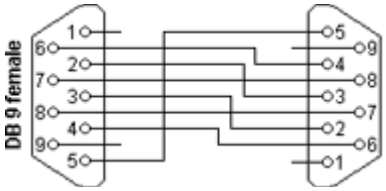
Zapojenie najčastejšie používaných káblov

3-vodičové prepojenie



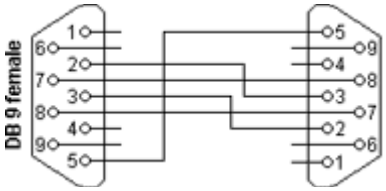
Obr. 7 3-vodičové zap.

7-vodičové prepojenie s úplným riadením toku



Obr. 8 7-vodičové zap.

5-vodičové prepojenie s riadením toku



Obr. 9 5-vodičové zap.

Vzájomné prepojenie sériových káblov s plným riadením toku

9pin D-Sub to 9pin D-Sub

DB9-1			DB9-2
Receive Data	2	3	Transmit Data
Transmit Data	3	2	Receive Data
Data Terminal Ready	4	6+1	Data Set Ready + Carrier Detect
System Ground	5	5	System Ground
Data Set Ready + Carrier Detect	6+1	4	Data Terminal Ready
Request to Send	7	8	Clear to Send
Clear to Send	8	7	Request to Send

25pin D-Sub to 25pin D-Sub

DB25-1			DB25-2
Receive Data	3	2	Transmit Data
Transmit Data	2	3	Receive Data
Data Terminal Ready	20	6+8	Data Set Ready + Carrier Detect
System Ground	7	7	System Ground
Data Set Ready + Carrier Detect	6+8	20	Data Terminal Ready
Request to Send	4	5	Clear to Send
Clear to Send	5	4	Request to Send

9pin D-Sub to 25pin D-Sub

DB9			DB25
Receive Data	2	2	Transmit Data
Transmit Data	3	3	Receive Data
Data Terminal Ready	4	6+8	Data Set Ready + Carrier Detect
System Ground	5	7	System Ground
Data Set Ready + Carrier Detect	6+1	20	Data Terminal Ready
Request to Send	7	5	Clear to Send
Clear to Send	8	4	Request to Send

Dĺžka vedenia RS232

Čo ovplyvňuje prenosovú rýchlosť a dĺžku vedenia?

Štandard RS-232 uvádza ako maximálnu možnú dĺžku vodičov 15 metrov alebo dĺžku vodiča o kapacite 2500 pF. To znamená, že pri použití kvalitných vodičov je možné dodržať štandard a pri zachovaní menovitej kapacite predĺžiť vzdialenosť až na cca 50 metrov.

Kábel je možné taktiež predlžovať pri znížení prenosovej rýchlosti, pretože potom bude prenos odolnejší voči veľkej kapacite vedenia. Uvedené parametre počítajú s prenosovou rýchlosťou 19200 Bd.

Texas Instruments uvádza ako výsledok pokusných meraní nasledujúce dĺžky vodičov v závislosti na prenosovej rýchlosti. Vzhľadom k "laboratórnym" podmienkam tohto merania je treba brať tieto údaje iba ako orientačné. V praxi je treba počítať s rušením atď.

Maximálna dĺžka vedenia

Baud rate [Bd]	Max length [ft]	Max length [m]
19 200	50	15
9 600	500	150
4 800	1 000	300
2 400	3 000	900

Pre prenos dát na väčšie vzdialenosti je výhodnejšie používať rozhranie RS-422, RS-485, či prúdovú slučku.

Čo je Baud?

Baud je jednotka používaná pre meranie rýchlosti prenosu dát. Prenosová rýchlosť definuje rýchlosť prenosu dát z dátového média na iné dátové médium. Baud rate udáva počet zmien signálu za sekundu. Počet zmien sa potom vyjadruje v baudoch. Ako základná jednotka informácie v moderných počítačových systémoch sa berie jeden bit (nadobúda hodnoty 0 alebo 1). Do jednej signálovej zmeny je možné zakódovať i viacej než jeden bit. A preto nie je možné zlučovať pojem bps (bits per second = bity za sekundu) s pojmom baud.
Jednotka baud je pomenovaná po Jean-Maurice-Émile Baudotovi (*1845 - †1903).

Parametre dátového prenosu

Parita

Parita je najjednoduchší spôsob ako bez nárokov na výpočtový výkon zabezpečiť prenos dát. Vo vysielacom zariadení sa spočíta počet jednotkových bitov a doplní sa paritným bitom tak, aby bola zachovaná vopred dohodnutá podmienka párneho alebo nepárneho počtu jednotkových bitov.

- **PÁRNA PARITA** – Počet jednotkových bitov + paritný bit = PÁRNE ČÍSLO,
- **NEPÁRNA PARITA** – Počet jednotkových bitov + paritný bit = NEPÁRNE ČÍSLO,
- **SPACE PARITY** – Tzv. nulová parita – paritný bit je vždy v log. 0, používa sa napríklad pri komunikácii 7-bitového zariadenia s 8-bitovým, kedy paritný bit nahradzuje tvrdú log. 0, posledný bit v bajte, tým je zachovaná kompatibilita s 8-bitovým prenosom,
- **MARK PARITY** – Paritný bit je nastavený tvrdo na log. 1, pri kompenzácii 7-bitového prenosu je treba ho na prijímacej strane nulovať, inak nie je kompatibilný s ASCII.

7-bitový/8-bitový formát

Na starých termináloch IBM, ktoré sa používali iba ako textové konzoly, ušetrili návrhári jeden bit prenosu a používali iba 7-bitový prenos, ktorý umožňoval 128 kombinácií. Dnes sa v praxi prakticky nepoužíva, ale stal sa štandardom.

STOP bit/bity

STOP BIT – Definuje ukončenie rámca. Zároveň zaisťuje určité oneskorenie pre prijímač. Práve v dobe prímu STOP bitu väčšina zariadení spracováva prijatý BAJT.
ZDVOJENÝ STOP BIT – Používa sa u pomalších zariadeniach pre dobeh spracovávania prijatého znaku. Jedná sa o štandard na 110 Bd .

Riadenie toku dát (HANDSHAKING)

Čo je handshaking

Riadenie toku dát (HANDSHAKING) predstavuje potvrdenie príjmu dát či pripravenosť k prenosu a jeho zahájenia na úrovni hardvérového alebo softvérového rozhrania.

Hardvérový handshaking

Prenos od vysielača k prijímaču, že vysielač má pripravené platné dáta k odoslaniu.
Prenos od prijímača k vysielaču, že prijímač je schopný dáta spracovávať.

Softvérový handshaking

Prebieha na úrovni komunikačných protokolov (ZMODEM, KERMIT...), pomocou bežného dátového kanálu prijímač vysielaču odkáže, či je schopný dáta prijímať a spracovávať. Dos/BIOS v počítačoch PC používa pre SW handshaking znaky v Ascii tabuľke XON/XOF. Ak je však potreba v toku dát znaky XON/XOF vyslať, je nutné vyslať špeciálnu sekvenciu znakov, čo samozrejme prenos dát obsahujúci prevažne tieto znaky značne spomalí.

SYCHRÓNNY a ASYNCHRÓNNY prenos dát

Čo je handshaking

SYCHRÓNNY prenos informácií znamená, že na niektorom vodiči alebo vodičoch sa nastaví určitá úroveň, ktorá prenáša informáciu a validita (kontrola) informácie sa potvrdí impulzom, alebo zmenou úrovne synchronizačného signálu. Synchronizačným signálom sa teda informácie kvantujú.

Základné vlastnosti SYCHRÓNNEHO prenosu:

- výhodné pre veľké objemy dát, prenášané po viacej vodičoch,
- je potrebné jednoznačne určiť, kto vysiela synchronizačné impulzy,
- možno použiť spojite premenlivú rýchlosť prenosu, napríklad podľa pomeru chybovosti,
- nutnosť synchronizačného vodiča "na viac" – v podstate "neprenáša žiadnu informáciu",
- na strane zariadenia nepotrebuje nijako zložitú elektroniku.

Hardvérový handshaking

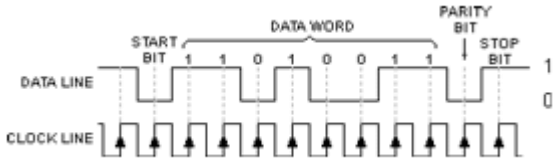
ASYNCHRONNÝ prenos dát prenáša dáta v určitých sekvenciách. Dáta sú prenášané presne danou rýchlosťou a štartovacou sekvenciou, na ktorú sa synchronizujú všetky prijímacie zariadenia. Všetky strany obsahujú vlastný presný oscilátor, vďaka ktorému odčítajú dáta v presne definovaných intervaloch. Po ukončení sekvencie je ďalší príjem opäť synchronizovaný štartovacou sekvenciou.

Základné vlastnosti ASYNCHRÓNNEHO prenosu:

- nevýhodné pre veľké objemy dát, ale vhodné pre dlhé vedenia, na nich by synchronizačný vodič činil nezanedbateľné finančné náklady,
- je možné použiť pre komunikáciu medzi mnohými zariadeniami,
- nutné definovať jednoznačne prenosové rýchlosti, zmenu rýchlosti je treba ošetriť softvérovou sekvenciou, ktorá prijme počítač zmeniť hardvérovú prenosovú rýchlosť,
- celkom zložitá a drahá elektronika, je potrebné použiť kryštálové oscilátory,
- až o 20 % menšia prenosová rýchlosť užitočných dát pri rovnakej rýchlosti komunikácie, vzhľadom k nutnosti štartovacích a paritných bitov.

Asynchronous Serial Data Frame (8E1)

RS-232 používa asynchrónny prenos informácií. Každý prenesený bajt konštantnou rýchlosťou je preto treba synchronizovať. K synchronizácií sa používa zostupná hrana tzv. Štart bitu. Za ňou už nasledujú posielané dáta.



Obr. 10 Asynchrónny prenos

Synchronizácia RS-232

RS-232 používa asynchrónny prenos informácií. Každý prenesený bajt konštantnej rýchlosti je preto treba synchronizovať. K synchronizácií sa používa zostupná hrana tzv. Štart bitu. Za ňou už nasledujú posielané dáta.

Obr. 11 Synchrónny prenos

RÁMEC – Prenosový rámec

Kompletná prenosová skupina = prenášané DÁTA (7/8-bitové) doplnená o ŠTART BIT, STOP BIT a PARITU. Prenosový rámec je teda minimálne prenášaná skupina dát.

Dočasné zastavenie dátového prenosu

Pokiaľ je treba upozorniť zariadenie na dočasné zastavenie vysielania, vygeneruje vysielač nepretržitý impulz v log. 0 po dobu 100–600 ms (maximálna doba linky v neprerušenej log. 0 je na najnižšej rýchlosti, keď sa vysiela 8xlog. 0 je 66,6 ms).

Hardvér

Obsluha portov BIOSom

INT 14H ROM-BIOS-u bude fungovať so všetkými štyrmi portami, teda ak uložíme bázové adresy týchto portov do tabuľky COM portov začínajúce na adrese 0:0400. Je treba, aby žiadne dva adaptéry nezdieľali rovnakú adresu, lebo ani jeden z nich nebude fungovať. BIOS ale podporuje len pomerne primitívnu komunikáciu využívajúcu stavové dotazy (polling), čo je pre zložitejšie aplikácie takmer nepoužiteľné. Adaptér je však schopný vyvolať hardvérové prerušenie na základe mnohých rôznych podmienok v závislosti na hodnotách registra povolenia prerušenia (3f9H alebo 2f9H).

Port	I/O adresa	IRQ	INT vektor
COM1	3f8H až 3ffH	4	0ch
COM2	2f8H až 2ffH	3	0bh
COM3	3e8H až 3efH	4	0ch
COM4	2e8H až 2ef8H	3	0bh

Radiče portov

Zoznam v BIOS-u obsahuje zoznam až štyroch bázových adries COM portov. Behom POST BIOS testuje a inicializuje COM1 a COM2. Tieto porty sú tvorené obvodmi 8250 u PC/XT alebo 16450 u PC/AT (rovnako programovo i vývodovo kompatibilné verzie). Nevýhodou týchto obvodov je, že sú schopné si zapamätávať iba jeden prijatý znak, takže môže hlavne pri vyšších prenosových rýchlostiach dôjsť k strate dát (overrun), keď počítač nestihne včas odobrať prijatý znak pred príchodom ďalšieho. Z tohto dôvodu bol vyvinutý obvod 16550, opäť vývodovo i programovo kompatibilný, ktorý však má 14 znakovú prijímaciu vyrovnávaciu pamäť. Táto pamäť však musí byť programovo zapnutá, inak sa obvod chová ako štandardný 16450. Je teda možno ho v komunikačných kartách vymeniť, štandardne je použitý v systémoch PS/2 a v portoch osadených na MB. Vie ho obsluhovať napr. známy komunikačný program Telix alebo Terminate, rovnako ako WINDOWS (možno vypnúť).

3f8H – Obsluha UART-u

Používa sa pre bežné čítanie a príjem znakov zo sériového kanálu. Pokiaľ je nastavený port 3FB, bit 7=1 (OUT 3fbH,80H), prečítame na tomto portu dolný bajt deliteľa, ktorý spoločne s horným bajtom (port 3f9H) tvorí 16-bitovú hodnotu, ktorá určuje prenosovú rýchlosť podľa tabuľky.

Rýchlosť	konštanta	3F9	3F8
110	1040	04	17
150	768	03	00
300	384	01	80
600	192	00	c0
1200	96	00	60
2400	48	00	30
4800	24	00	18
9600	12	00	0c
19200	6	00	6
115200	0	00	00

3fcH – Riadenie modemu

7 MSB	6	5	4	3	2	1	0 LSB

SET - nastaví do log. 1
CLR - nastaví do log. 0
Log. 1 = -3..15V na pinu
Log. 0 = -3..15V na pinu

- bit 0:1 = SET DTR (data terminal ready),
- bit 1:1 = SET RTS (request to send),
- bit 2:1 = SET OUT1 (užívateľský výstup),
- bit 3:1 = SET OUT2,
- bit 4:1 = aktivácia spätnej väzby pre diagnostické účely.

Bity OUT1 a OUT2 sú z hľadiska výrobcu príslušného integrovaného obvodu vývody na puzdre, ich funkcia je k dispozícií návrhárom systému (proste dva programom ľahko ovládateľné bity pre ľubovoľné upotrebenie). Návrhári IBM PC využili bit OUT2, ktorý priviedli na vstup povolenia prerušenia obvodu. Keďže v pokojovom stave je v bitu OUT2 zapísaná 0, je prerušenie ZAKÁZANÉ (blokovaním na úrovni hardvér), i keď správne naprogramujeme register povolenia prerušenia a všetko ostatné. **Pokiaľ chceme používať prerušenie, musíme OUT2 nastaviť na 1.**
Táto poznámka je len v niektorých manuáloch drobnými písmenami a keď by sa spočítali všetky hodiny, počas ktorých si vývojári lámali hlavy, prečo im to nie a nie prerušovať. Skóre autorov SYSMAN-u je 14 dní.

3feH – Register stavu modemu

7 MSB	6	5	4	3	2	1	0 LSB

BIT	Skratka	Popis
0	DCTS	Delta Clear To Send – CTS zmenil stav.
1	DDSR	Delta Data Set Ready – DSR zmenil stav.
2	TERI	Trailing Edge Ring Indicator – vzostupná hrana indikátora vyzváňania.
3	DDCD	Delta Data Carrier Detect – DCD zmenil stav.
4	CTS	Clear To Send (CTS) je aktívny.
5	DSR	Data Set Ready (DSR) je aktívny.
6	RI	Ring Indicator (RI) je aktívny.
7	DCD	Data Carrier Detect (DCD) je aktívny.

Myš a jej parametre

Klasická sériová myš sa pripojuje na RS-232. Hardvérový protokol však nie je úplne štandardný a líši sa podľa toho, či používame 3tlačítkovú myš – **Mouse System Mouse** alebo myš **Microsoft**, ktorá vie obsluhovať iba **2 tlačidlá** a používa 3-bajtový prenos dát, zatiaľ čo 3-tlačítková myš používa obecne 5-bajtový.

Pripojenie RS-232 na TTL

Trochu z histórie

Ak používame v zariadení TTL alebo CMOS obvody, budeme musieť ich logickú RS-232 linku napäťovo upraviť pred pripojením do PC, pretože napäťové úrovne RS-232 nie sú priamo zlučiteľné zo žiadnou logikou.
Pre toto upravenie sa štandardne používali obvody 1488 a 1489, ktoré ale potrebovali + 12 V a −12 V pre vytvorenie výstupných úrovní. To bolo mimochodom jedným z dôvodov, prečo je v klasickom PC zo zdroja vyvedené i −12 V a −5 V (ďalším dôvodom bola potreba väčšieho rozdielového napätia u historických dynamických pamätí pre zvýšenie ich rýchlosti).

MAXIM

Prielom v tomto smere urobila firma MAXIM svojím obvodom MAX232. Využila totiž svoje znalosti vo vývoji spínaných nábojových meničov napätia a vyvinula obvod, ktorý vystačil s +5 V a potrebné napätie si samostatne vyrobil pomocou 4 externých kondenzátorov. Obvod samozrejme konvertuje log. 0 na +3,15 V a log. 1 na −3,15 V, ako je popísané vyššie.
MAX232 sa stal neuveriteľným šlágrom a dnes jeho obdobu nájdeme takmer vo všetkých komerčných zariadeniach pripojovaných k RS-232.

Katalógové listy

Katalógový list MAX232
Katalógový list ICL232 – Alternatíva k MAX232 od Intersilu
Katalógový list DS14C232 – Alternatíva k MAX232 od National Semiconductors
Katalógový list MAX201
Katalógový list MAX 233
Katalógový list MAX3232

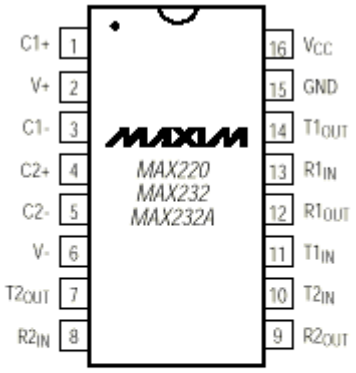
MAX 232 – Prevodník RS-232/TTL MAX 232

MAX 232 - Prevodník RS-232/TTL MAX 232

Jedná sa o prevodník TTL na RS-232. Obsahuje dve dvojice oddeľovačov konvertujúcich napäťové úrovne. Napätie pre RS-232 sa získava pomocou nábojovej pumpy a výstupné napätie preto značne závisí na kvalite použitých kondenzátorov, ktoré u elektrolytických kondenzátorov časom značne klesá. Napätie je možné získať na pinoch 2 a 6 a použiť pre ďalšie obvody.
Obvod funguje vždy na prvé zapojenie. Maxim vyrába i verzie s minimálnou externou kapacitou – (MAX 232A – 0,1 µF) alebo verzia pracujúca v rozsahu 7,5–13 V (určené pre batériové aplikácie) – MAX 201 a MAX 231. Špecialitou firmy MAXIM sú obvody MAX203 a MAX 233, ktoré dokážu pracovať úplne bez potreby externých kondenzátorov.

UPOZORNENIE: *Vzhľadom k úspešnosti MAX232 začalo mnoho firiem vyrábať obvody pinovo kompatibilné v nižšej cenovej hladine. U jedného z týchto výrobcov (myslím AD232), ktoré u nás svojho času predávalo GM, je potreba opačne polarizovať jeden z elektrolytov, tak uvádza firemný katalógový list. Vzhľadom k predpokladanej kompatibilite to však mnoho vývojárov neoveruje, a potom vznikajú časom veľmi komplikované poruchy. Odporúčame preto používať buď originálne obvody MAXIM, alebo dobre preštudovať "substitučné" obvody vzhľadom k predpokladaným odlišnostiam.*

MAX 232 – Zapojenie vývodov

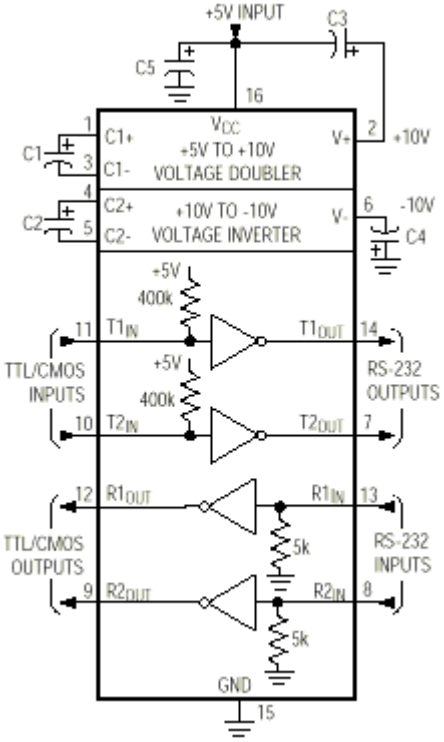


DIP/SO

CAPACITANCE (µF)					
DEVICE	C1	C2	C3	C4	C5
MAX220	4.7	4.7	10	10	4.7
MAX232	1.0	1.0	1.0	1.0	1.0
MAX232A	0.1	0.1	0.1	0.1	0.1

Obr. 12 MAX 232 – Zapojenie vývodov

MAX 232 – Aplikačná schéma



Obr. 13 MAX 232 – Aplikačná schéma

Prevodníky rozhrania RS-232 – prenos dát na väčšiu vzdialenosť

Prečo používať prevodníky?

Rozhranie RS-232 je oficiálne (podľa špecifikácie) možné použiť pre prepojenie dvoch zariadení medzi sebou a to len do vzdialenosti **15 metrov pri prenosovej rýchlosti do 20 kb/s**, čo vyplýva z povolenej kapacity káblov 2500 pF.
V praxi sú dosahované výsledky oveľa lepšie (115200 kb/s pri vzdialenosti až 50 metrov) vďaka použitiu káblov s kapacitou pod 1000 pF.
Rozhranie RS-232 je relatívne málo odolné proti rušeniu, lebo prenos dát je realizovaný napäťovou úrovňou na vodičoch (voči GND) na zaťažovacom odpore 3,7 kΩ pri šumovej imunite 3 V. Veľa zariadení má ale vstupnú impedanciu oveľa vyššiu (až 30 kΩ) a šumovú imunitu nižšiu (1 V), takže dochádza k zvýšenému rušeniu a tým k zmenšenému možnému dosahu linky. V každom prípade sa odporúča použiť tienený kábel a venovať pozornosť spôsobu prevedenia signálovej zeme a zeme zariadenia (v plnej špecifikácií RS-232 sú to dva samostatné vodiče).

Prenos dát na väčšiu vzdialenosť

Pre prepojenie dvoch zariadení s rozhraním RS-232 v minimálnej konfigurácií stačia tri vodiče (RxD, TxD, Gnd – tzv. null modem kábel), potom sa ale nevyužívajú riadiace signály (RTS, CTS, DSR, DTR a ďalšie). Pri prepojení riadiacich signálov počet vodičov rastie podľa požiadavky na spôsob riadenia toku dát. Pre prenos dát na väčšie vzdialenosti sa používa rozhranie **RS-422** alebo **RS-485**, prípadne **prúdová slučka**.

Prenos dát pomocou RS-422 a RS-485

Fakticky sa jedná o diferenciálnu prúdovú slučku, kde dátové stavy vyjadruje smer tečúceho prúdu v samostatnom páre vodičov pre každý komunikačný smer.
Podľa špecifikácie je dosah týchto rozhraní **1200 metrov a prenosová rýchlosť 10 Mb/s**. Toho je možné dosiahnuť na vedení tienenou **krútenou dvojlinkou** (u RS-422 na dvojvodiči) a **ukončením vedenia** zakončovacími odpormi **120 Ω** na oboch koncoch vedenia.

Prúdová slučka

Čo je prúdová slučka?

Pre komunikáciu na väčšie vzdialenosti sa občas používa taktiež prúdová slučka 0/20 mA (pozor, nezamieňať so slučkou 0(4) až 20 mA pre prenos analógových veličín).
Prúdová slučka je vysoko odolná proti rušeniu, lebo slučkou buď tečie alebo netečie prúd 20 mA. Prevod je relatívne jednoduchý a dosah sa rádovo zvýši na stovky metrov, zatiaľ čo komunikačná rýchlosť zostane zachovaná. Z ekonomických dôvodov sa na prúdovú slučku prevádzajú iba signály RxD a TxD z plného rozhrania RS-232. Pokiaľ potrebujeme prenášať ešte nejaké signály (RTS, CTS/DTR, DSR), použijeme 2 kanálovú verziu prevodníka.

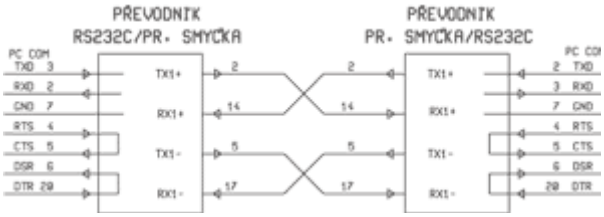
Dĺžka vedenia a galvanické oddelenie

Dĺžka vedenia je obmedzená kapacitou a ohmickým odporom vedenia. V praxi je rozhodujúcim faktorom len odpor. U bežne používaných prúdových slučiek býva povolený odpor vedenia do 200 Ω. Prúdové slučky sa od seba líšia prevedením vysielačov a prijímačov. Môžu byť aktívne alebo pasívne, priame alebo galvanicky oddelené.
Galvanické oddelenie môže byť i tým hlavným dôvodom, ktorý vedie k použitiu prúdovej slučky. Vedenie prúdovej slučky sa realizuje veľmi jednoducho, v minimálnom prípade stačia štyri vodiče, ktoré ani nemusia byť skrútené.

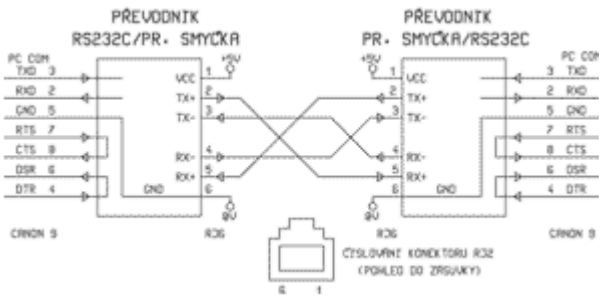
Použitie prúdovej slučky

Prepojuje sa vzájomne vždy jeden vysielač s jedným prijímačom a to v kombinácií aktívny vysielač - pasívny prijímač alebo pasívny vysielač - aktívny prijímač.

Zapojenie prúdovej slučky



Obr. 14 Zapojenie prúdovej slučky 1



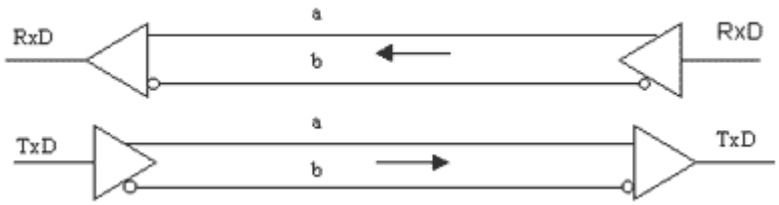
Obr. 15 Zapojenie prúdovej slučky 2

Rozhranie RS-422

Čo je RS-422?

Linka RS-422 používa jeden pár vodičov pre signál RxD a druhý pre signál TxD. Z toho vyplýva, ak použijeme linku RS-422 k predĺženiu prenosovej vzdialenosti miesto "trojvodičovej" RS-232 (RxD, TxD, GND), nič sa nemusí na spôsobe komunikácie meniť a nie treba ani zásah do softvér.
Každý zo signálov linky je prenášaný po dvojici vodičov, najlepšie v prevedení twistový pár. Vodiče označované **a** a **b** sú vysielačom budené v proti fáze a prijímač vyhodnocuje ich napäťový rozdiel. Týmto princípom sa odstránia súčtové (aditívne) rušenia.
Linky môžu byť vedené až na vzdialenosť 1600m (vodiče s kapacitou do 65pF/m) a je možné ich vetviť.

Schéma zapojení linky RS-422



Obr. 16 Linka RS-422

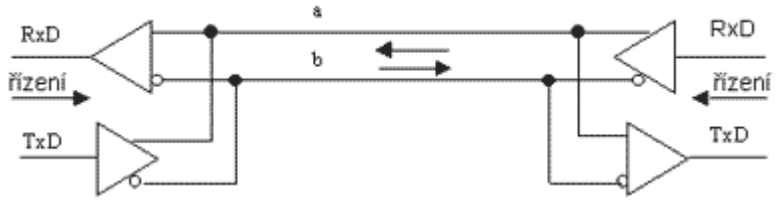
Viacej o rozhraní RS-422 na: http://www.manualy.sk/poucha.html

Rozhranie RS-485

Čo je RS-485?

Na rozdiel od linky RS-422 používa rozhranie RS-485 len jeden pár vodičov pre oba smery toku dát. Je teda treba smer komunikácie prepínať a to môže byť problém zvlášť v prípadoch, keď s touto možnosťou softvér nepočíta.
Prepínanie smeru komunikácie iste bude vyriešené u zariadenia, ktoré obsahuje už štandardne linku RS-485. Pokiaľ však používame zariadenie s vyvedenou linkou RS-232 (napríklad počítač PC) a následným prevodníkom RS-232/RS-485, je treba prepínanie smeru zaistiť. Najvhodnejší spôsob je použiť pre prepnutie niektorý voľný riadiaci signál linky RS-232 (napríklad DTR alebo RTS), jeho ovládanie však musí umožniť použitý program (pozor pri písaní takého programu pre PC – vďaka rôznym bufferom na nových motherboardoch to nie je úplne jednoduché).

Schéma zapojenia linky RS-485



Obr. 17 Linka RS-485

Viacej o rozhraní RS-485 na: http://www.manualy.sk/poucha.html

Softvér

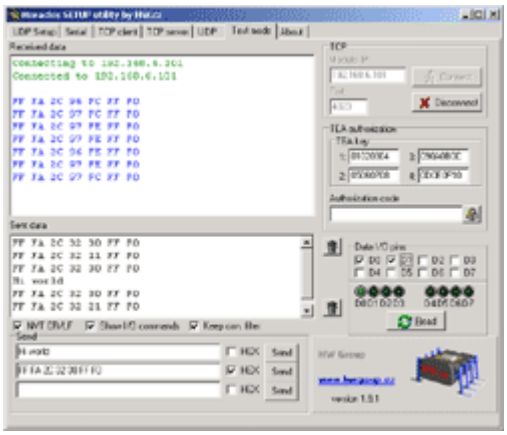
Hercules SETUP utility

Hercules SETUP utility je užívateľský terminál pre sériovú linku (RS-232 alebo RS-485), UDP/IP a TCP/IP (klient alebo server). S originálnym HW Ethernet zariadením (Serial/Ethernet Converter, RS-232/Ethernet Buffer alebo I/O Controller) môže byť používaný i pre nastavenie parametrov týchto zariadení pomocou UDP. Keďže bol Hercules pôvodne vytvorený len pre interné potreby HWGroup, je dnes šírený ako freevér.

General useful parts:

- **Serial port terminal** – podpora COM5 a vyšších,
- **TCP/IP Client terminal**,
- **TCP/IP Server** "**terminal**",
- **UDP** "**terminal**".

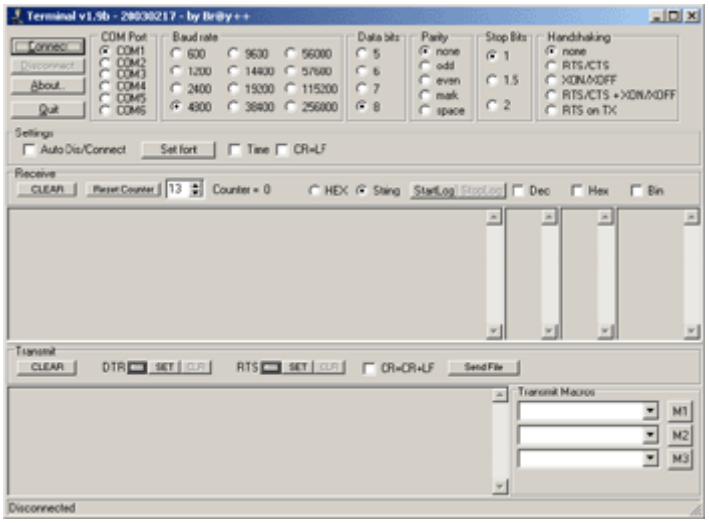
Download:
HWgroup.cz – http://www.hw.cz/projects/www.hwgroup.cz/default.htm
Heracles SETUP Utility –
http://www.hw.cz/projects/www.hwgroup.cz/download/HerculesSetup.zip



Obr. 18 Hercules SETUP utility

Terminal

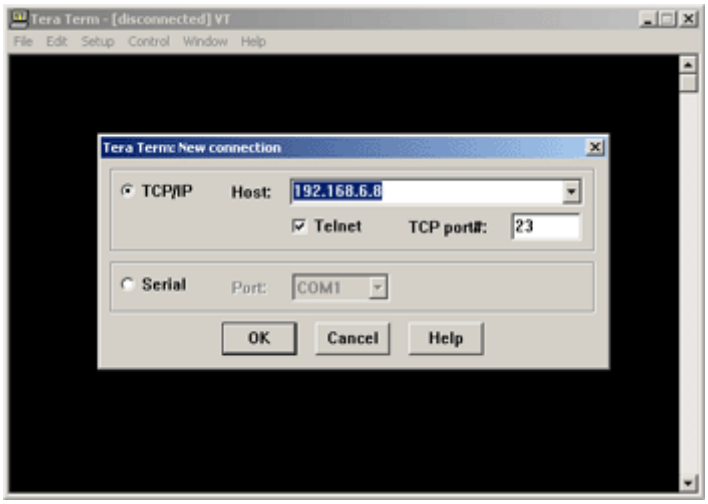
Terminal je jednoduchý terminálový program pre sériový port (COM). Môže byť použitý pre komunikáciu rôznych zariadení, ako sú modemy, routery, embedded µC systémy, GSM telefóny.
Download:
The Terminal – http://www.hw-server.com/priloha/termv19b.zip



Obr. 19 Terminal

Tera Term

Tera Term (Pro) je terminálový program pre MS-Windows. Podporuje VT100, telnet, serial port, a ďalšie.
Download:
Tera Term Pro ver. 2.3 for Windows 95/NT – http://www.hw-server.com/priloha/ttermp23.zip (943,376 bajtov)



Obr. 20 Tera Term

Ovládače, utility, programovanie

Ovládanie RS-232 pomocou PPP.exe

K pokusom so sériovým portom, hľadanie porúch, jednoduchému ladeniu RS-485 a podobným účelom je určený program PPP.EXE (Papouchův Pomocný Program), ktorý uľahčí jednoduché pokusy so sériovou linkou. Programom je možné vysielať a prijímať jednotlivé znaky, meniť stavy signálov linky RS-232 a ovládať prepínanie smeru komunikácie.
Download programu a kompletný článok:

Ovládače na sériový port pod WINDOWS

Pokiaľ programujeme ovládací softvér pod Windows (95, 98, NT, XP) k akémukoľvek zariadeniu, máme veľký problém s prístupom na porty, lebo Microsoft konečne začal používať definíciu operačného systému z roku 1960 a u Windows NT operačný systém skutočne stráži, aby užívateľský SW nerobil, čo nemá na HW vrstve. Preto je pri programovaní akýchkoľvek HW zariadení potreba pristupovať k týmto zariadeniam korektne cez Windows API.
Download rutiny a kompletný článok:

Sériová komunikácia RS-232 - programový model

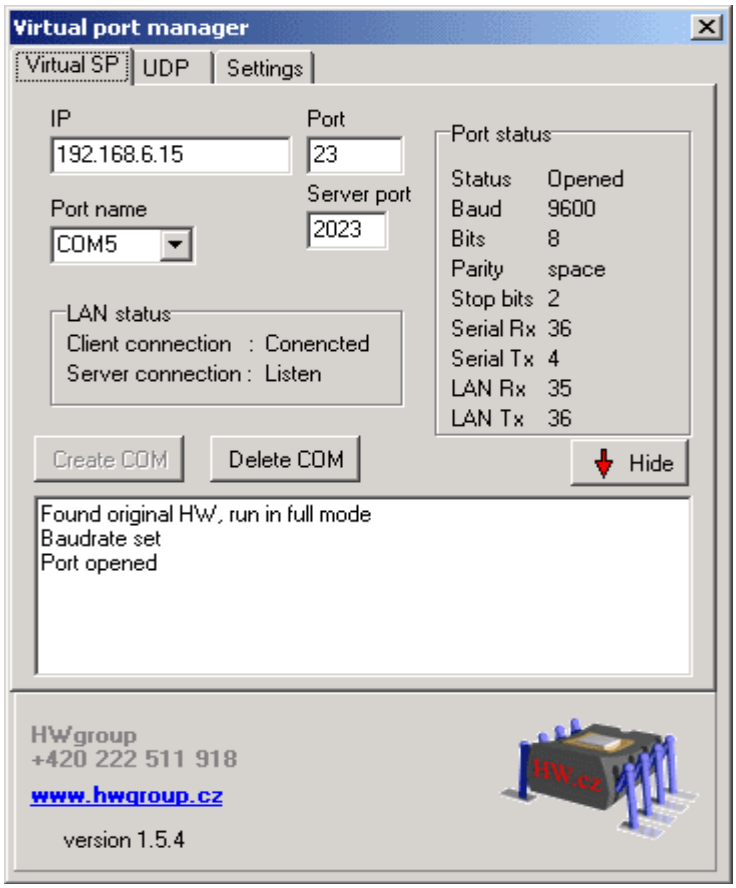
Tu si môžeme stiahnuť .PDF dokument, ktorý na jednej stránke zhrňuje všetko, čo potrebujeme vedieť k programovaniu sériového rozhrania. Popis všetkých registrov, postup vyvolávania prerušenia atď.
Všetko v jednom súbore s dĺžkou cca 13 kB.
Download - RS232_prog_model.pdf:

Virtuálne porty

HW Virtual Serial Port

Virtuálny driver sériového portu pre Windows je softvérový prostriedok, ktorý pridá do operačného systému zdanlivý sériový port, napríklad COM 5 a dáta z tohto portu presmeruje na iné hardvérové rozhranie. V dnešnej dobe sa virtuálny port využíva hlavne pre pripojenie sériového rozhrania RS-232 cez USB alebo po sieti Ethernet.
HW Virtual Serial Driver je primárne určený pre potreby www.hw.cz, ale je možné ho zadarmo použiť ako univerzálny driver pre vytvorenie vzdialeného virtuálneho sériového portu, ktorý dáta presmeruje na definovanú TCP/IP adresu a port.
PC s nainštalovaným driverom (ovládačom) sa na viac vie chovať i ako server, takže spojenie môže inicializovať i zariadenie zaslaním akýchkoľvek dát do vzdialeného portu. Konvertor na základe dát z RS-232 otvorí spojenie s PC a predá ho virtuálnemu comu. Celá situácia sa tak maximálne blíži štandardnému riešeniu s klasickým sériovým portom.

Tento virtuálny driver vie fungovať ako client i ako server!
DOWNLOAD:
HW Virtual Serial Port – http://www.hw.cz/download.php3?HW_VirtualSerialPort.zip (4 Mb)
Manuál k HW VSP –
http://www.hw.cz/download.php3?hw_vsp_v104.pdf (341 kb)
Anglická verzia manuálu –
http://www.hw.cz/download_en.php3?hw_vsp_v104_en.pdf (282 kb)



Obr. 21 HW Virtual Serial Port

SPY (monitoring)

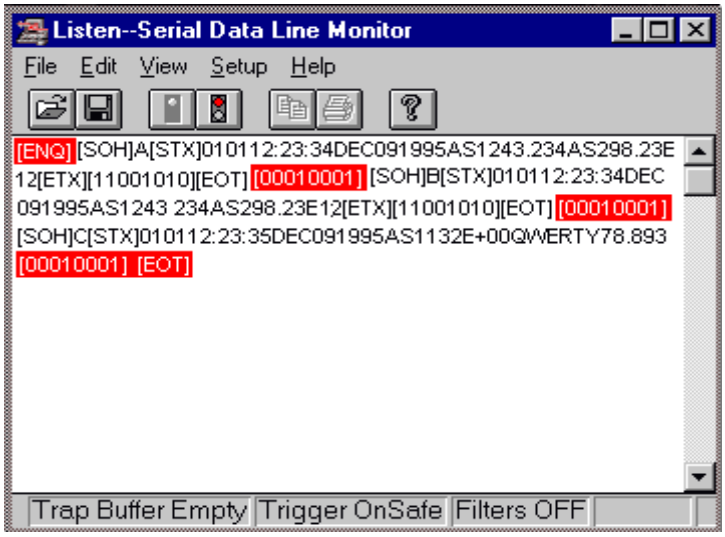
Listen

Listen je program fungujúci ako monitor dátového spojenia navrhnutý k odstraňovaniu chýb a problémov na sériovej linke. Využíva štandardný sériový port počítača (COM) pre snímanie obojsmernej komunikácie medzi dvoma zariadeniami.
Listen je program fungujúci ako monitor dátového spojenia navrhnutý k odstraňovaniu chýb a problémov na sériovej linke. Využíva štandardný sériový port počítača (COM) pre snímanie obojsmernej komunikácie medzi dvoma zariadeniami.

Základné vlastnosti:

- dáta sú zachytávané a zobrazované v reálnom čase. Je možné ľahko definovať užívateľské druhy písma a farby prostredia,
- na dátový prenos je možné aplikovať filtre umožňujúce izolovať chybový stav,
- veľkou výhodou Listenu je jeho **schopnosť zobraziť netlačiteľné znaky** v rôznych formátoch nastavených užívateľom,
- vysoko prispôsobivý program je ideálnym riešením pre prácu v "teréne" v spojení s napríklad notebookom.

DOWNLOAD:
WinTECH Software – http://www.win-tech.com/html/listen.htm
Listen – http://www.hw-server.com/priloha/listen32.zip (307K)



Obr. 22 Listen

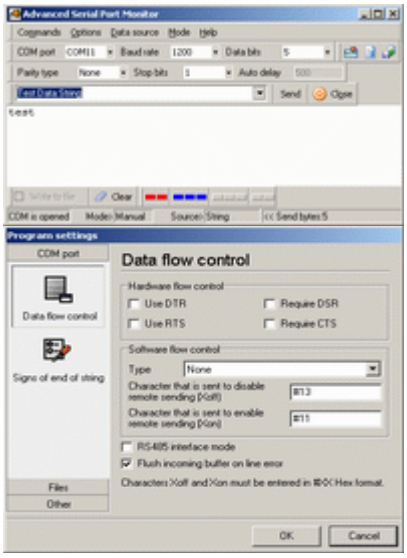
Advanced Serial Port Monitor

Advanced Serial Port Monitor je program, ktorý môže byť použitý ku kontrole dátového toku pomocou sériového portu počítača.

Základné vlastnosti:

- plne duplexná prevádzka,
- pružné nastavovanie parametrov,
- ukladanie prijímaných dát do súboru,
- automatický a manuálny režim prevádzky,
- špionážny (spy) režim.

DOWNLOAD:
AGG Software – http://www.aggsoft.com/products/supercom/
Advanced Serial Port Monitor – http://www.aggsoft.com/download/aspmon30.exe (1318 kB)



Obr. 23 Advanced Serial Port Monitor

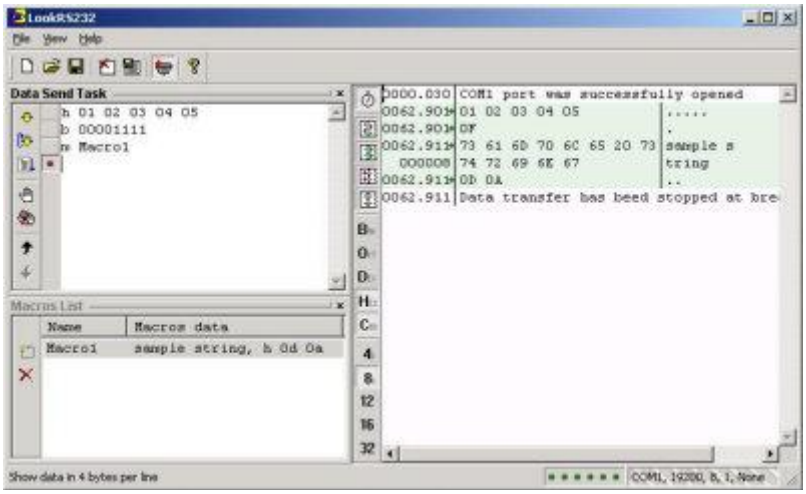
Ladenie (Debuging)

Look RS232

Look RS232 je nástroj pre ladenie komunikácie počítača s periférami pripojenými na sériový port ako sú modemy, mini-ATS, projektory a podobne. Jednoduché grafické rozhranie umožňuje ľahkú prácu so sériovým portom. Look RS232 podporuje spojenie pri štandardných rýchlostiach komunikácie (110…115200 kbit/s). Sú podporované rôzne dátové formáty ako ASCII, BIN, HEX, OCT a zaznamenáva priebeh komunikácie spolu s časom.

Základné vlastnosti:
- záznam prichádzajúceho i odchádzajúceho dátového toku s možnosťou ukladania do súboru,
- trasovanie, prerušenie a ladenie krok za krokom,
- zobrazenie stavu komunikačného portu,
- užívanie makier k zautomatizovaniu často vykonávaných operácií,
- úplná história prichádzajúcich a odchádzajúcich dát a príkazov,
- časová informácia pre prichádzajúce i odchádzajúce dáta a udalosti,
- prenosy dát zo súborov,
- plne duplexný,
- podpora virtuálnych sériových portov cez USB alebo Ethernet.

DOWNLOAD:
fCoder Group – http://www.fcoder.com/
Look RS232 – http://www.fcoder.com/support/go.php?id=009
Look RS232 – http://www.hw-server.com/priloha/lookrs232.zip



Obr. 24 Look RS232

DOWNLOAD & Súvisiace odkazy

Katalógové listy

Katalógový list MAX232
Katalógový list ICL232 – Alternatíva k MAX232 od Intersilu
Katalógový list DS14C232 – Alternatíva k MAX232 od National Semiconductors
Katalógový list MAX201
Katalógový list MAX 233
Katalógový list MAX3232
Katalógový list HSDL-7000
Katalógový list HSDL-1001 – výroba ukončená
Katalógový list HSDL-3612 – pokračovateľ HSDL-1001
Katalógový list FT8U232AM
Katalógový list FT232BM
Katalógový list FT2232C

Konvertory RS-232 / RS-422 / RS-485 / Ethernet / USB

Rôzne prevodníky z rozhraní RS-232 na RS-422/485/Prúdová slučka/USB/Ethernet
Prevodníky sériových rozhraní na optické vlákno
RS-232/485 – ETHERNET konvertor
RS-232 » Prúdová slučka 20mA
RS-232 » RS-232 – galvanické oddelenie
RS-232 » RS-422
RS-232 » RS-485
RS-232 » USB
RS-232 » USB
RS-232 » Ethernet
WiFi / RS-232
WiFi / 2xRS-232

Ovládače a ďalšia dokumentácia

Ovládače sériového portu pre Windows
Sériová komunikácia RS-232 - Programový model
PPP – program pre ovládanie jednotlivých pinov RS-232 a pre vysielanie a príjem signálov.
Podrobný popis registrov COM-ových portov – rs232_pc_registry.zip

RS232/TTL KONVERTOR

Zatiaľ, čo klasické sériové rozhranie RS-232 pracuje s napäťovými úrovňami ± 12 V, jednočipové mikroprocesory a mnohé pomocné komunikačné obvody pracujú s úrovňami TTL a pre pripojenie k PC je treba ich konvertovať pomocou prevodníkov úrovní. A pokiaľ ten nie je súčasťou skúšaného obvodu, prichádzajú ku slovu externé obvody, ako je RS-232/TTL Konvertor.

RS-232/TTL Konvertor je jednoduché zariadenie v podobe sériového (COM) káblu určeného pre občasné pripojenie jednočipových aplikácií k PC. Poskytuje užívateľovi možnosť ľahkého testovania sériovej komunikácie u hotových aplikácií alebo ich častí pracujúcimi s úrovňami TTL.

Základné vlastnosti

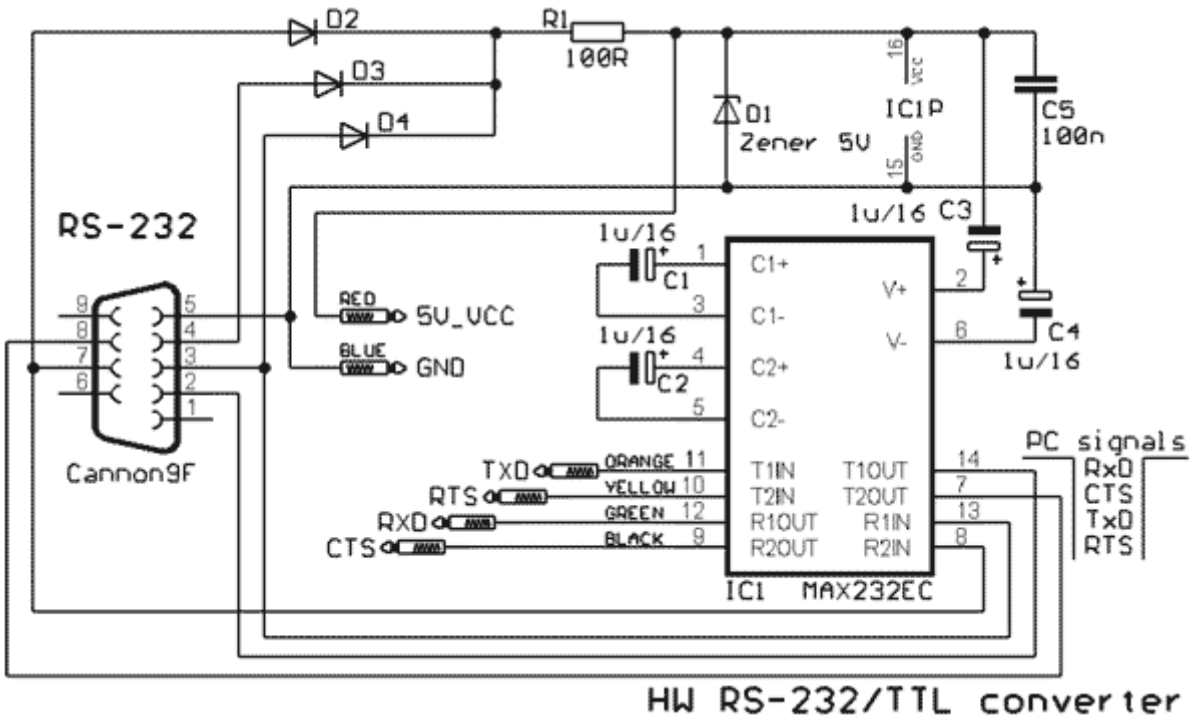
- konverzia signálov RxD, TxD, RTS, CTS,
- možnosť hardvérového handshakingu,
- pripravený k okamžitému použitiu,
- prevedenie káblu na strane PC zakončeného zásuvkou DB-9 (Cannon),
- TTL strana prevodníka zakončená vodičmi,
- možnosť napájania nízko príkonových zariadení priamo z COM portu PC.

Príklady použitia

- testovanie zariadení, komunikujúcich po sériovej linke
- ladenie sériovej komunikácie,
- priame pripojenie mikroprocesora k PC,
- ovládanie LCD displejov cez sériový port,
- pripojenie teplomerov k PC,
- pripojenie mobilného telefónu či PDA k PC.

Obr. 25 RS232/TTL Konvertor

Základom prevodníka je notoricky známy integrovaný obvod MAX232, ktorý konvertuje napäťové úrovne rozhrania RS-232 (-3 až -15 V a +3 až +15 V) na logické signály 5 V logiky TTL. Zatiaľ, čo RS-232 strana konvertoru obsahuje konektor DB-9 pre priame pripojenie do COM portu PC, na TTL strane sú vyvedené len konce prepojovacieho káblu, čo umožňuje použitie ľubovoľného konektoru či priame zapojenie do aplikácie.



Obr. 26 WH RS-232/TTL konvertor

Upozornenie!

Prevodník úrovní je treba napájať z vyvedených vonkajších 5 V, alebo zaistiť +3 až 15 V aspoň na výstupu DTR alebo RTS!

PRENOS DÁT PO LINKÁCH RS-485 A RS-422

Pre prenos dát medzi zariadeniami sa často používa sériová komunikácia. Zatiaľ, čo snaha pre zrýchlenie toku dát (napríklad medzi jednotlivými obvodmi v jednom prístroji) vedie k užívaniu synchrónneho prenosu (dáta, synchronizačné impulzy, rámec), pre malé objemy dát a väčšie vzdialenosti je naopak výhodná asynchrónna komunikácia. Pojem "malý objem dát" je tu myslená rýchlosť v rádoch 1 až 100 kb za sekundu. Asynchrónna komunikácia minimalizuje počet vodičov potrebných k prenosu, čím sa zlacňuje komunikačné vedenie.

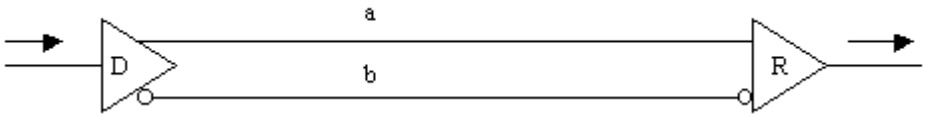
Linky RS-232, RS-485 a RS-422

Komunikácia po linke RS-232 je najbežnejšia, pretože rozhranie RS-232 má vyvedený každý bežný počítač. Používa sa pre pripojenie zariadení komunikujúcich maximálnou rýchlosťou 115,2 kBd na vzdialenosť maximálne 15 m. Okrem vodičov pre prenos dát - RxD a TxD obsahuje ešte ďalšie vodiče pre riadenie toku dát. Tieto pomocné riadiace signály nie sú obsiahnuté v linkách typu RS-422 ani RS-485 a musia byť nahradené komunikačným protokolom. Ani mnohé zariadenia komunikujúce po linke RS-232 tieto signály nevyužívajú.

Nevýhodou linky RS-232 je obmedzená komunikačná vzdialenosť a nemožnosť jeho vetvenia. Navyše obvykle nebýva od zariadenia galvanicky oddelené, čo prináša problémy so zemnými slučkami, ktoré v priemyselnom prostredí celú komunikáciu znemožnia. Preto tam, kde nie je možné použiť inú linku, je možné odporučiť aspoň galvanicky oddeliť všetky používané signály. Okrem zlepšenia komunikácie sa tak predíde k zničeniu budičov a prijímačov zariadení.

Pre prenos dát na väčšie vzdialenosti je vhodné použiť linku RS-485 alebo RS-422. Linky môžu byť vedené až na vzdialenosť 1600 m (vodiče s kapacitou do 65pF/m) a je možné ich vetviť.

Každý zo signálov linky je prenášaný po dvojici vodičov, najlepšie v prevedení twistový pár. Vodiče označované a a b sú vysielačom budené v proti fázy a prijímač vyhodnocuje ich napäťový rozdiel. Týmto princípom sa odstránia súčtové (aditívne) rušenia.



Obr. 27 Prenos jedného signálu po linke RS-485 alebo RS-422. D je vysielač, R je prijímač

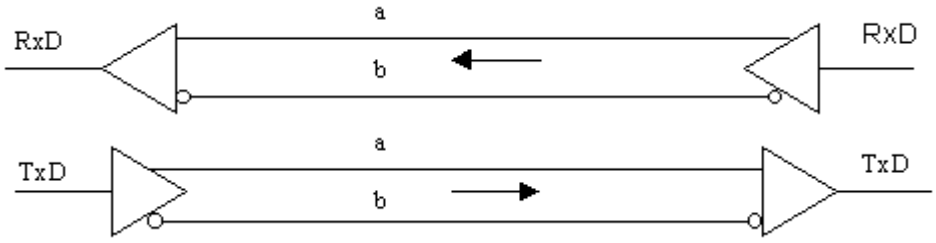
Z toho vyplýva i základné odporučenie pre prevedenie linky RS-485 alebo RS-422 – ak nie je k dispozícií skrútený (twistový) pár vodičov, je treba použiť aspoň tak vedené vodiče, aby sa do oboch indukovali poruchy zhodne.

Zatiaľ, čo linka RS-232 pracuje s úrovňami typicky –12 V a +12 V, úrovne linky RS-485 alebo RS-422 sú menšie, typický rozdiel medzi vodičmi je 2 V. Aby prijímač mohol pracovať diferenciálne, nesmie byť rozdiel medzi zemou vysielača a zemou prijímača väčší ako 7 V. V opačnom prípade sa vstupy prijímača zahltia a dôjde k prerušeniu komunikácie.

Preto je nutné používať linky RS-485 a RS-422 vždy s galvanickým oddelením, inak sa ich výhody stratia.

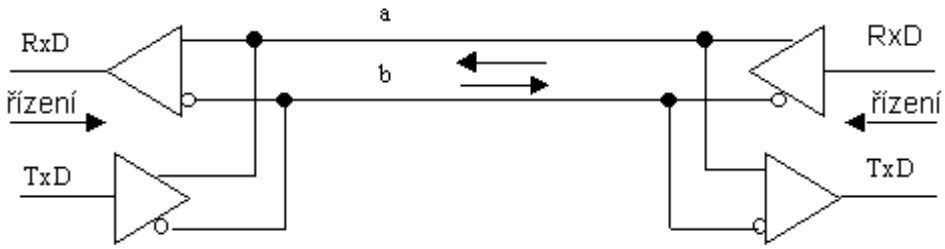
Prevedenie linky RS-485 a RS-422

Prevedenie oboch liniek je na obrázkoch 2 a 3. Ako už bolo uvedené, obe linky prenášajú iba dáta a nepoužívajú žiadne riadiace signály.



Obr. 28 Prevedenie nevetvenej linky RS-422

Linka RS-422 používa jeden pár vodičov pre signál RxD a druhý pre signál TxD. Z toho vyplýva, ak použijeme linku RS-422 k predĺženiu prenosovej vzdialenosti miesto ‘‘trojvodičovej’’ RS-232 (RxD, TxD, GND), nič sa nemusí na spôsobe komunikácie meniť a nie je treba ani zásah do softvéru.



Obr. 29 Prevedenie nevetvenej linky RS-485

Linka RS-485 používa jeden pár vodičov pre oba smery toku dát. Je treba smer komunikácie prepínať a to môže byť problém zvlášť v prípadoch, keď s touto možnosťou softvér nepočíta.

Prepínanie smeru komunikácie bude vyriešené u zariadenia, ktoré obsahuje už štandardne linku RS-485. Pokiaľ však používame zariadenie s vyvedenou linkou RS-232 (napríklad počítač PC) a následným prevodníkom RS-232/RS-485, je treba prepínanie smeru zaistiť. Najvhodnejší spôsob je použitie pre prepnutie niektorých voľných riadiacich signálov linky RS-232 (napríklad DTR alebo RTS), jeho ovládanie však musí umožniť použitý program (pozor pri písaní takého programu pre PC – vďaka rôznym bufferom na nových motherboardoch to nie je úplne jednoduché).

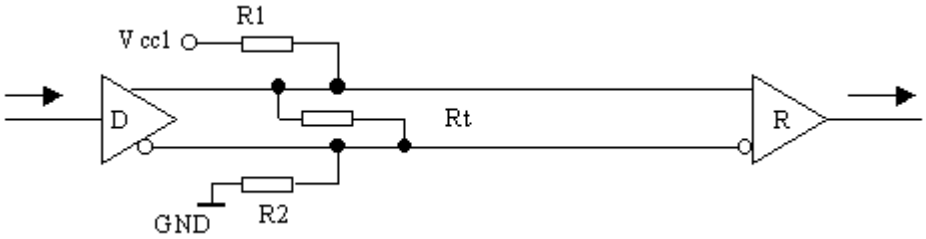
Ak nie je signál pre prepnutie k dispozícii, je jedinou možnosťou použiť prevodník linky RS-232 na RS-485 s automatickým prepínaním. I to má však úskalia. Takýto prevodník je stále prepnutý na priamo z linky RS-485 a pri zistení dát vysielaných zo strany linky RS-232 sa prepne na vysielanie. V režimu vysielania však prevodník zostane ešte po nejakú dobu (pretože nemôže presne identifikovať koniec dát). Ak behom tejto doby začne na linku vysielať niekto iný, dôjde ku kolízii a dáta nie sú prijaté.

Problémy s prepínaním je možné však po preskúmaní konkrétnej situácie takmer vždy vyriešiť. Poslednou možnosťou je použiť linku RS-422, ktorá prepínanie nepotrebuje. U rozvetvených liniek môže byť počet zariadení na linke maximálne 16, avšak existujú prijímače s menšou záťažou, takže ich môže byť až 128. Linka by mala byť prevedená ako linka s krátkymi odbočkami, nie ako strom alebo hviezda.

Zakončenie zbernice

Impedančné zakončenie linky RS-485 alebo RS-422 je vecou dosť problematickou. Samozrejme je správne na konci linky zapojiť rezistor o rovnakej hodnote s impedanciou vedenia a tým zabrániť odrazom na vedení. V praxi však nie sú často používané vysoké rýchlosti prenosu (typické sú 9,6 kBd alebo 19,2 kBd) a ani vedenie nebýva správne prevedené. Zakončenie potom stráca zmysel a len znižuje úroveň signálu a tým i odolnosť proti poruchám. Preto je vhodné voliť zakončenie skôr väčšie, do 1000 Ohmov.

Oveľa dôležitejšie ako impedančné zakončenie je definovanie pokojového stavu linky. Pretože pri komunikácií po linke RS-485 alebo pri rozvetvenej linke RS-422 sa vysielače odpojujú, dochádza k dobám, keď na linku žiadne zariadenie nevysiela. V tejto dobe nie je stav linky definovaný a linka je extréme citlivá na indukované napätia (poruchy), ktoré sa javia ako prichádzajúce dáta. Preto je treba definovať kľudový stav linky pripojením rezistorov podľa obrázku (predpokladáme, že v kľude je vodič b zápornejší ako a).



Obr. 30 Definovanie pokojového stavu linky

Odpor Rt je zakončovací (napr. 150 Ohmov), odpory R1 a R2 definujú pokojový stav (oba asi 470 Ω až 1 k Ω). Vcc a GND sú lokálne napájané a zem budiča.

Čo je vodič a a čo je vodič b?

Rozpoznanie vodičov linky RS-485 alebo RS-422 nie je zložité, ale v značení je neporiadok. Vodiče a a b bývajú rôzne značené u rôznych výrobcov a ani z normy EIA nie je zrejmý ich vzájomný potenciál v pokojovom stave. Pokiaľ teda označenie vodičov nie je jasné, je jediným riešením uviesť zariadenie do pokojového stavu pri vysielaní a polaritu zmerať, alebo prehodením správne zapojenie nájsť. Prehodením vodičov nie je možné budiče linky zničiť, priamo norma predpisuje prúdové obmedzenie.

Protokoly

V histórií bola rada pokusov o zavedenie štandardných komunikačných protokolov na sériových linkách, avšak žiadny z nich sa neujal. Typy komunikácie sú štandardné maximálne pre zariadenia od jedného výrobcu. Táto skutočnosť komplikuje pripojenie viacej rôznych zariadení na jednu linku, pretože dochádza ku kolízií dát.

Prúdová slučka

Pre komunikáciu na väčšie vzdialenosti sa najskôr používala taktiež prúdová slučka 0/20 mA (pozor, nezameniť so slučkou 0(4) až 20 mA pre prenos analógových veličín). Aj jej vlastnosti sú však u vyšších rýchlostí horšie ako u liniek RS-485 a RS-422, preto pre nové zariadenia nie je vhodná a v praxi sa príliš nepoužíva.

Záver

Linky RS-485 alebo RS-422 je možné odporučiť pre najrôznejšie prenosy dát v priemyselnom prostredí. Pri správnom prevedení je spoľahlivosť prenosu v porovnaní s linkou RS-232 alebo prúdovou slučkou vysoká.

Návrh experimentálneho pracoviska na výučbu mikroprocesorovej techniky

MARIÁN PLAČKO

21.06. 2005

Úvod

- Základnou myšlienkou diplomovej práce bolo vytvoriť experimentálne vývojové pracovisko s mikrokontrolérmi RISC, ktoré by spĺňalo nasledujúce požiadavky:
  - Jednoduchosť konštrukcie
  - Finančnú nenáročnosť
  - Rýchly vývoj a testovanie
  - Pedagogické využitie

- V komplexnejších projektoch je ťažisko pri samotnej realizácii v oblasti návrhu a testovania hardvéru. Z tohto dôvodu je dôležité:

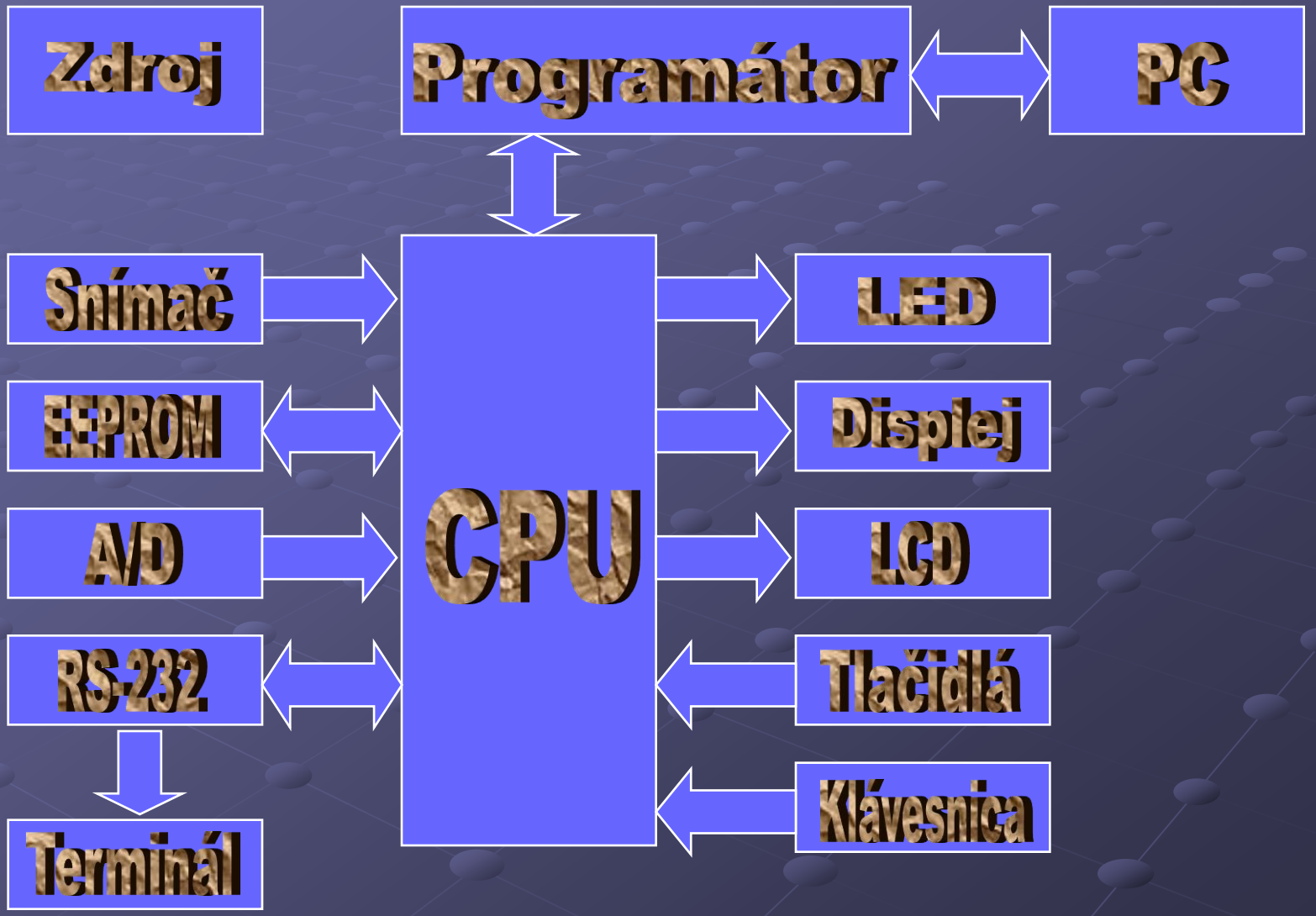
  - Vhodná voľba mikrokontroléra
  - Vhodná voľba vývojových a podporných nástrojov
  - Nasadenie vyšších programovacích jazykov napr.: C, Visual Basic, Java, a pod.

Materiál a metódy

- Rodina mikrokontrolérov – ATMEL AVR:
  - High-speed CMOS technológia
  - FLASH pamäť programu
  - ISP programovanie
  - RISC inštrukčný súbor
  - Harvardská architektúra

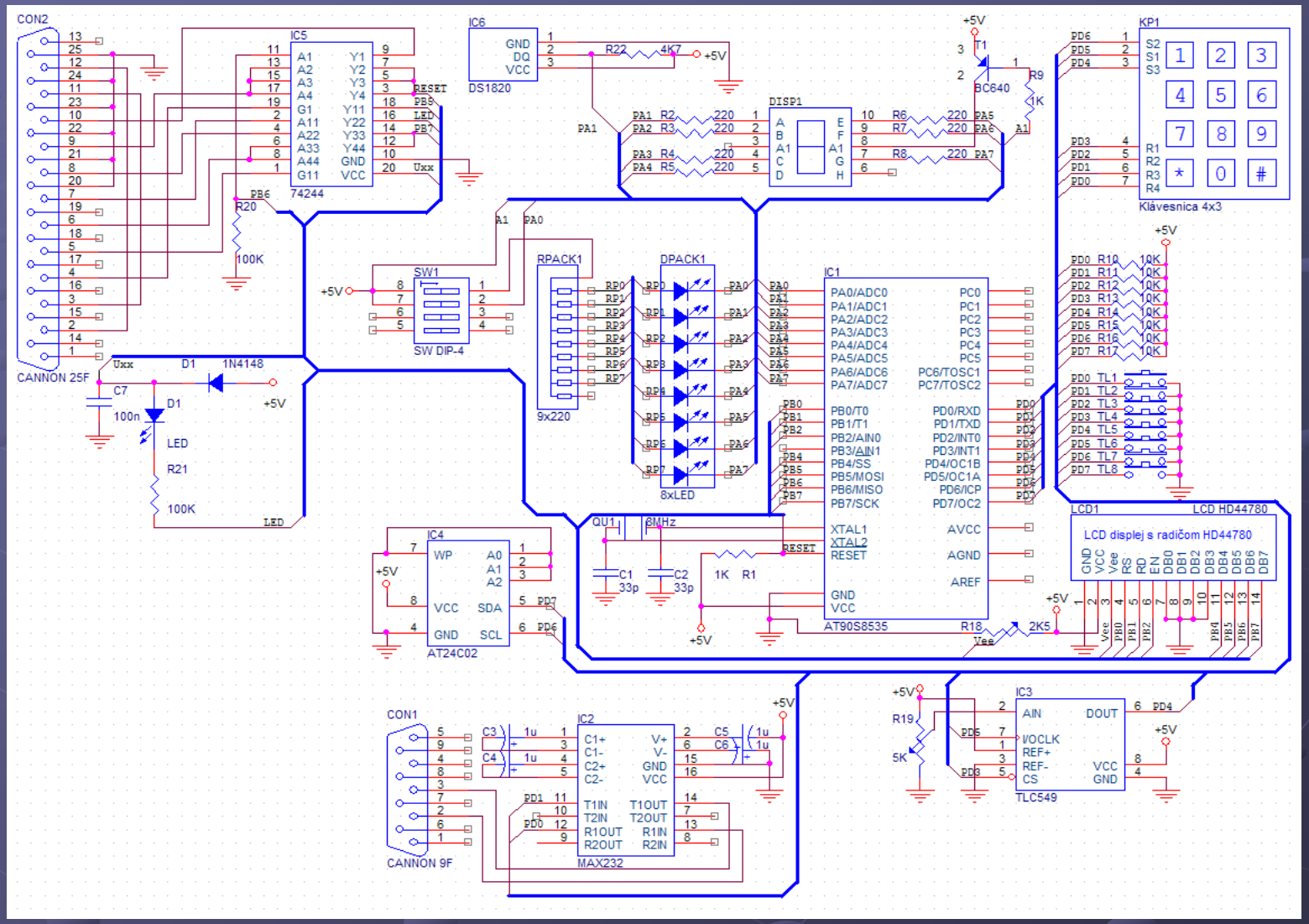
- Typ mikrokontroléra – AT90S8535:
  - FLASH: 8 kB
  - SRAM: 512 B
  - EEPROM: 512 B
  - Maximálna frekvencia: 8 MHz
  - Počet inštrukcií: 118
  - Počet registrov: 32
  - Čítač/časovač: 1x8 bit, 2x16 bit
  - Watchdog
  - A/D prevodník: 8 kanálový 10 bitový
  - Analógový komparátor
  - UART
  - SPI
  - I/O: 32 pinov
  - 1000 programovacích cyklov
  - ...

- Bloková schéma:

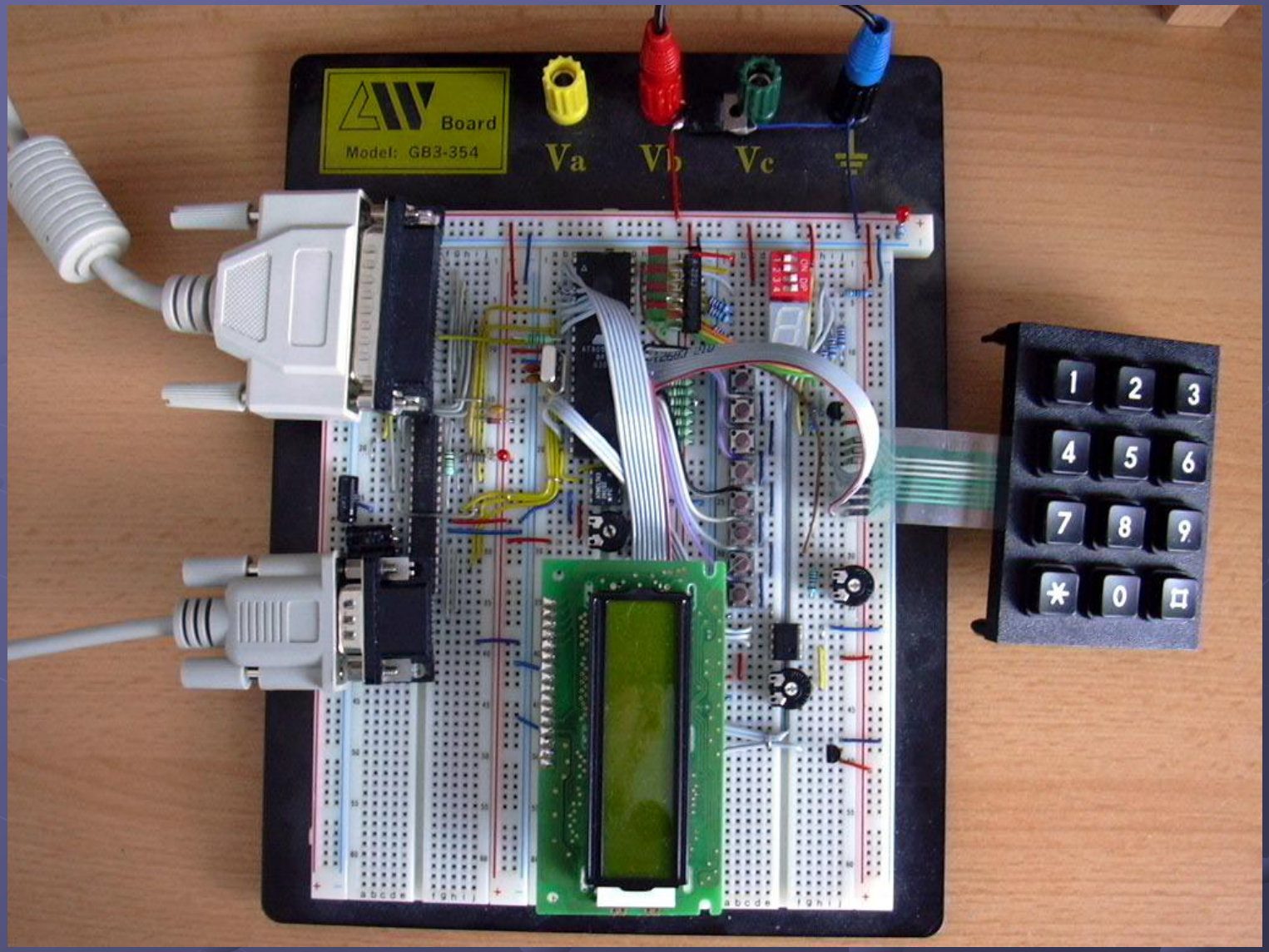


Obr. 1 Bloková schéma

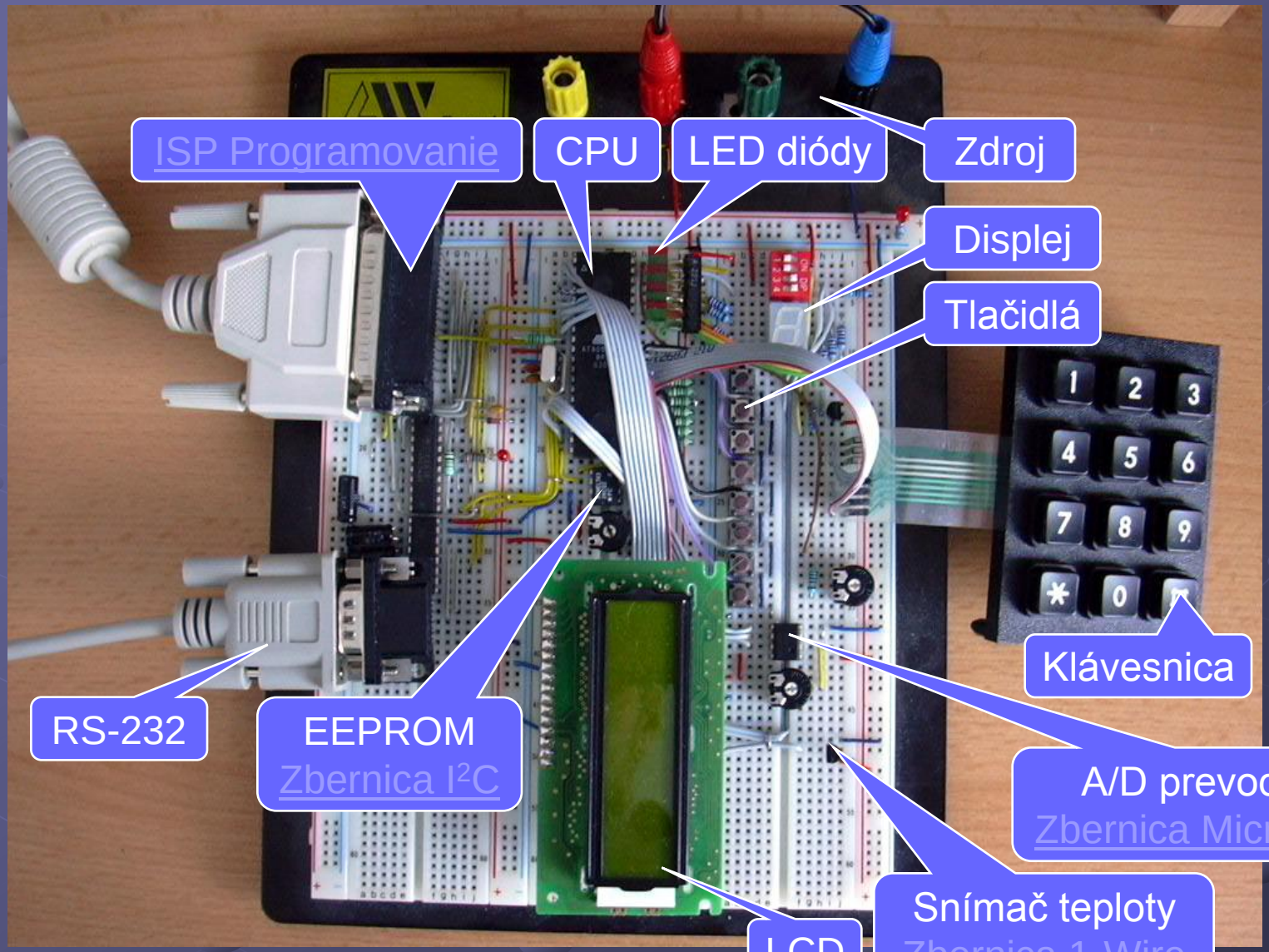
Výsledky



Obr. 2 Schéma zapojenia



Obr. 3 Experimentálne pracovisko



Obr. 4 Funkčné časti

Návrhy na využitie

ISP Programovanie

- **ISP** *(In-System Programming)* – metóda, ktorá umožňuje programovanie obvodov priamo v zariadení pomocou vyhradených vodičov určených na komunikáciu pri programovaní.
- **SPI** *(Serial Peripheral Interface)* – rozhranie pre sériovú synchrónnu komunikáciu.

Späť na "Funkčné časti"

Zbernica MicroWire

- Výrobca: National Semiconductor
- Použitie: EEPROM, A/D, atď.
- Zbernica je tvorená 3 vodičmi:
  - CLK *(Clock)* – hodinový signál, riadi prenos
  - SO/DI *(Serial Out/Data In)* – spája výstup mikrokontroléra so vstupmi periférnych obvodov
  - SI/DO *(Serial In/Data Out)* – spája výstupy periférnych obvodov na vstup mikrokontroléra
- Zbernica nedefinuje formát prenášaných dát

Poznámka:

- CS *(Chip Select)* – výber periférneho obvodu

Späť na "Funkčné časti"

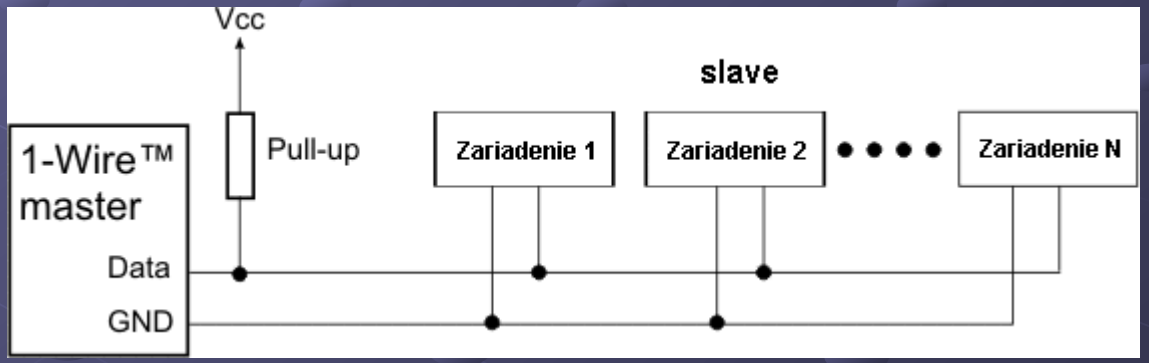
Zbernica I^2C

- I^2C *(Inter-Integrated-Circuit Bus)*
- Výrobca: Philips
- Použitie: EEPROM, A/D, D/A, PLL, RTC, atď.
- Zbernica je tvorená 2 vodičmi:
  - SCL – prenáša hodinový signál
  - SDA – synchrónny prenos dát
- Zbernica I^2C nedefinuje formát prenášaných dát
- Počet podriadených obvodov: $2^3 = 8$

Späť na "Funkčné časti"

Zbernica 1-Wire

- Výrobca: Dallas Semiconductor
- Použitie: Snímače, EEPROM, atď.
- Zbernica je tvorená 1 vodičom:
  - Data − riadi komunikáciu, prenos dát
- Slave − ROM, 64 b unikátne číslo



Obr. 5 Komunikácia po zbernici 1-Wire

Späť na "Funkčné časti"

Návrhy na využitie

- Ovládanie LED diód
- Komunikácia s LCD displejom
- Vyslanie reťazca znakov na port RS-232
- Snímanie vstupov z tlačidiel
- Maticová klávesnica
- A/D prevodník – voltmeter
- Zbernica I^2C – AT24C02 (sériová pamäť)
- Zbernica MicroWire – TLC549 (meranie napätia)
- Zbernica 1-Wire – DS18B20 (meranie teploty)
- Ovládací panel (ovládanie vybraných aplikácií – OS)

Meranie teploty

```
#asm
    .equ __w1_port = 0x1b
    .equ __w1_bit = 1
#endasm
```

Inicializácia zbernice 1-Wire

```
#asm
    .equ __lcd_port = 0x18
#endasm
```

Inicializácia LCD displeja

```
#include <lcd.h>
#include <ds1820.h>
#include <delay.h>
#include <math.h>
#include <stdio.h>

char lcd_buffer [33];

/* Maximálny počet DS1820/DS18B20 zariadení pripojených na 1-Wire zbernicu. */
#define MAX_DEVICES 8

/* Uloženie ROM kódu zariadení DS1820/DS18B20. */
unsigned char rom_code [MAX_DEVICES,9];
```

Inicializácia na uloženie ROM kódu

```
main ()
{
  unsigned char i, j, devices;
  int temp;

  lcd_init (16);
  lcd_putsf ("CodeVisionAVR\n1 Wire Bus");
  delay_ms (2000);
  lcd_clear ();

  /* Detekcia počtu zariadení DS1820/DS18B20 pripojených na zbernicu 1-Wire. */
  devices = w1_search (0xf0, rom_code);
  sprintf (lcd_buffer, "%u DS1820\nDevice detected", devices);
  lcd_puts (lcd_buffer);
  delay_ms (2000);

  /* Zobrazenie ROM kódu pre všetky zariadenia. */
  if (devices)
  {
    for (i = 0; i < devices; i ++)
    {
      sprintf (lcd_buffer, "Device #%u ROM\nCode is:", i + 1);
      lcd_clear ();
      lcd_puts (lcd_buffer);
      delay_ms (2000);
      lcd_clear ();
      for (j = 0; j < 8; j ++)
      {
```

Detekcia snímačov na zbernici 1-Wire

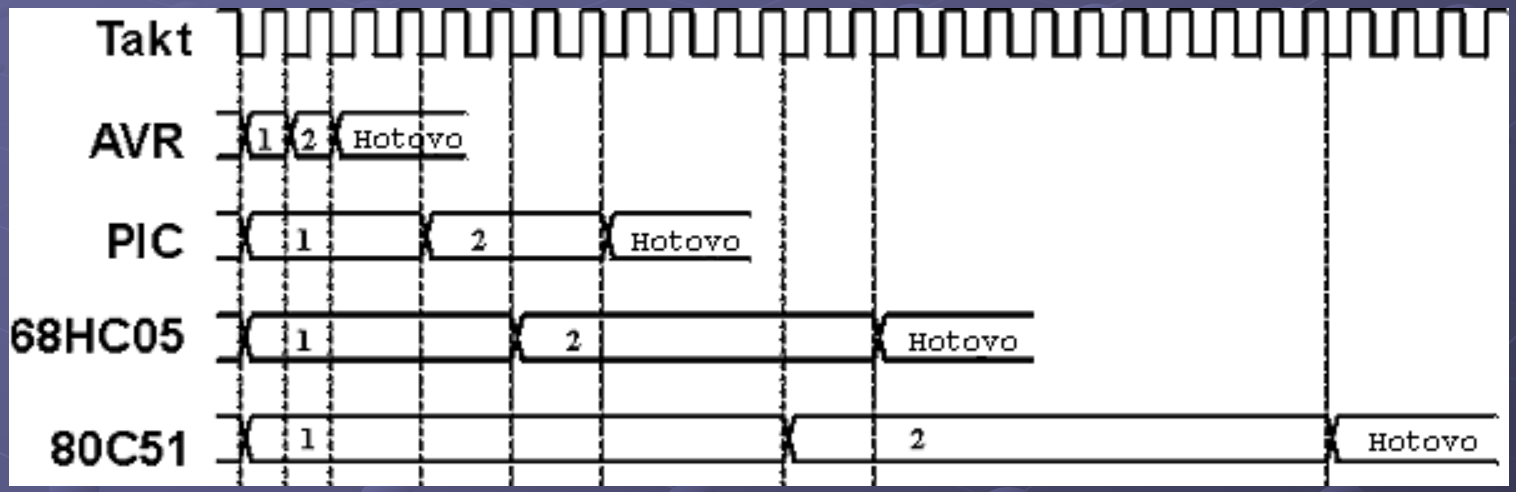
Zobrazenie ROM kódu snímačov

```
      sprintf (lcd_buffer, "%02X ", rom_code [i, j]);
      lcd_puts (lcd_buffer);
      if (j == 3) lcd_gotoxy (0, 1);
     };
    delay_ms (5000);
   };
}
else
while (1);  /* Stop, ak sa nenašlo žiadne zariadenie DS1820/DS18B20. */

/* Meranie a zobrazovanie teploty. */
while (1)
{
  for (i = 0; i < devices;)
  {
    temp = ds1820_temperature_10 (&rom_code [i, 0]);
    sprintf (lcd_buffer, "t%u=%i.%u\xdfC", ++ i, temp / 10, abs (temp%10));
    lcd_clear ();
    lcd_puts (lcd_buffer);
    delay_ms (800);
  };
};
}
```

Meranie a zobrazovanie teploty

Diskusia

- Výkon (x51, AVR, PIC, Motorola)



Obr. 6 Porovnanie počtu potrebných taktov na vykonanie dvoch inštrukcií pre vybrané typy mikrokontrolérov

Pozn.: AVR AT90S8535 ➔ 8 MHz ➔ 8 MIPS

PIC 16F84 ➔ 32 MHz ➔ 8 MIPS (výrobca max. 20 MHz)

- Spotreba (AT90S1200 verzus PIC16F84)
  - RUN mód:
    - AVR − cca. 7,2 mA (4 MHz / 5,5 V)
    - PIC − typická spotreba 1,8 mA / maximálna 4,5 mA (4 MHz / 5,5 V)
  - IDLE mód:
    - AVR − 1,5 mA (4 MHz / 5,5 V)
    - PIC − nepodporuje
  - POWER DOWN:
    - AVR − Watchdog on: cca. 30 µA pri 4 V
Watchdog off: < 1 µA pri 4 V
    - PIC − Watchdog on: typická spotreba 7 µA / maximálna 28 µA pri 4V
Watchdog off: typická spotreba 1 µA / maximálna 16 µA pri 4 V

Záver

- Diplomová práca bola riešená s cieľom zvýšiť vedomostnú úroveň študentov a podať základný prehľad o problematike mikroprocesorovej techniky.
- Experimentálne pracovisko spolu s vhodným softvérovým riešením tvorí dohromady výkonný, cenovo i konštrukčne nenáročný nástroj na vývoj, testovanie a realizáciu zariadení na báze mikrokontrolérov ATMEL AVR.
- Na jej ďalšie využitie sa predpokladá, že si k nej užívateľ postaví patričné prevodníky, výkonové obvody a snímače podľa vlastných požiadaviek.

Ďakujem za pozornosť